Krisensimulation mit abstrakten Handlungstypen: ein neuer, methodischer Ansatz

Will, Dieter

Veröffentlichungsversion / Published Version
Dissertation / phd thesis

Krisensimulation mit abstrakten Handlungstypen: Ein neuer, methodischer Ansatz

MÜNCHEN 2000

Inaugural–Dissertation
zur Erlangung des Doktorgrades
der Logik und Wissenschaftstheorie
an der
LUDWIG–MAXIMILIANS–UNIVERSITÄT
MÜNCHEN

vorgelegt von

Dieter Will

Vorwort

In der vorliegenden Arbeit wird gezeigt, daß eine Simulation von Krisen mit abstrakten Handlungstypen möglich ist. Ein globaler, nichtmilitärischer Ansatz ist leicht zu verwirklichen. Schon mit geringem Aufwand können Ergebnisse erzielt werden. Es erscheint lohnenswert, diesen Ansatz intensiver zu verfolgen.

Im Abschnitt 1 wird allgemein der Nutzen von Computersimulationen diskutiert. Gerade im Bereich der Krisen, ist wegen der großen Datenmenge, der Computer ein unverzichtbares Werkzeug. In Abschnitt 2 werden Krisen definiert. Die bisherigen Computerprogramme sind meist darauf ausgerichtet aus realen Daten, Regeln zu extrahieren. Die Datenbanken und ihre Programme werden kurz vorgestellt. Danach folgen zwei Abschnitte über Handlungen. Die Einordnung von Handlungen bildet die Grundlage dieser Simulation, die unabhängig von empirischen Daten erfolgt. Trotzdem lassen sich die gewonnen Simulationsergebnisse mit empirischen Daten vergleichen. Dabei zeigen sich durchaus Übereinstimmungen.

Grundlage der Programmierung ist die Programmiersprache Prolog. Für soziale Simulationen ist Prolog als Programmiersprache am besten geeignet. Moderne Simulationen verwenden auch Java, Jini oder andere objektorientierte Sprachen. Die wichtigsten Bestandteile dieser Programmiersprache, soweit sie für das Verständnis des Programms notwendig sind, werden in Abschnitt 5 vorgestellt. Im nächsten Abschnitt werden verschiedene Simulationen kurz erläutert (Axelrods Simulation über „Enstehung neuer politischer Akteuere“, SMASS und DMASS).

In Abschnitt 7 ab Seite 84 wird KRIS (**Kris**en**S**imulation) ausführlich erklärt. Im Anhang finden sich die Tabellen über Krisen, welche die Grundlage der Programmierung bilden. Der gesamte Quellcode von KRIS ist abgedruckt. Graphische Darstellungen verdeutlichen die Ergebnisse der Simulation.

Danksagung

Herr Prof. Dr. Wolfgang Balzer hat mir die Erarbeitung der Promotion in vorliegender Aufgabenstellung ermöglicht. Dafür bin ich ihm zu großem Dank verpflichtet. Sein für mich entwickeltes Engagement war erheblich. Stets war er zu einem klärenden Gespräch bereit, ohne meine selbständige Arbeit einzuengen.

München, 20. März 2000 Dieter Will

Layout–Konventionen

Alle Elemente der Programmsprachen, Programme und der Inhalt von Dateien werden in Schreibmaschinenschrift dargestellt (Beispiel: `write/1`). Für Dateinamen selbst wird eine serifenlose Schrift benutzt (Beispiel: actions.prolog). Ein Begriff in spitzen Klammern ($< Begriff >$) darf nicht direkt eingegeben werden. Vielmehr ist der übergeordnete Begriff einzusetzen. Beispielsweise ist in `tell`$(<$ `Dateiname` $>)$ der konkrete Dateiname einzugeben. Englische Begriffe werden kursiv geschrieben (Beispiel: *guarded clause*). Für mathematische Symbole wird kursive Schrift verwendet, soweit sie nicht Bestandteil von Programmen sind (Beispiel: $a \leq b$).

Inhaltsverzeichnis

1 Simulationen

Simulationen und Spiele wurden schon in Frühzeiten der Menschheit genutzt, um die Spieler mit Situationen und Problemen zu konfrontieren, die in der Realität auftreten. Durch beide werden Lösungsmöglichkeiten eingeübt, um mit Krisensituationen umzugehen. Spiele sind meist Nullsummenspiele entgegengesetzter Parteien, die formalisierte Regeln haben. Simulationen sind Modelle der Realität mit wenig Variablen. Die Simulation ist ein simplifiziertes und begrenztes Modell einer komplexen realen Situation.
Der Computer ermöglicht weitreichende Simulationen von großer Komplexität. Durch sie können die Probleme eines Sachverhaltes leichter verstanden und analysiert werden. Simulation und Modellierung spielen eine wichtige Rolle in Begründungen und Problemlösungen. Oftmals sind Simulationen ein wichtiges, manchmal das einzige Werkzeug zur Formalisierung und Problemlösung im Umgang mit Systemen[1]. Ein Grund für die Verwendung von Simulationen zur Modellierung komplexer Systeme ist unzureichende Information. In sozialen Bereichen ist der Informationsmangel oftmals mit Schwierigkeiten der Formalisierung komplexer Systeme verbunden.

> „Da Analysen sozialer Systeme oft nicht ohne Eingriffe in soziale Prozesse möglich (Reaktivität) oder aus moralischen Gründen nicht vertretbar sind (Forschungsethik), bietet die Modellbildung einen Ausweg, um wissenschaftliche Untersuchungen auch ohne unmittelbare Einwirkung auf zu analysierende soziale Prozesse vorzunehmen. Indem Modelle einen wissenschaftlich interessanten Wirklichkeitsauschnitt abbilden, bereiten sie ihn in einer Weise auf, daß unter den als relevant erachteten Aspekten am Modell ähnliche Beobachtungen gemacht werden können wie in der Realität.“[2]

Theorien über die Realität können gerade im sozialen Kontext ziemlich komplex werden. Meist ist es leichter, eine Theorie zu einer Simulation zu entwickeln. Diese Theorie kann dann an der Realität geprüft werden.

> „Die durch Deduktion bzw. Simulation gewonnenen Modelldaten erlauben häufig Voraussagen über das Verhalten des realen Systems in der Zukunft und über die

[1] siehe Ivan Futo, [49], 1990, Seite 12.
[2] Andreas Engel, [43], 1995, Seite 41.

> Reaktionen des realen Systems auf Eingriffe von außen. Diese Voraussagen können dazu verwendet werden, um ausgehend vom abstrakten Modell handelnd in das reale System einzugreifen. Dabei darf aber nicht übersehen werden, daß die Modelldaten bis zu einem gewissen Grad fehlerbehaftet sind und daher die getroffenen Voraussagen nur mit einer gewissen Wahrscheinlichkeit auch für das reale System zutreffend sind.“[3]

Traditionell war die Computersimulation numerisch orientiert. Durch qualitativ höherwertige logische Werkzeuge sind auch nichtnumerische Betrachtungen möglich. Die Programmiersprache Prolog bietet dazu die besten Möglichkeiten.
In der Einleitungzu ihrem Buch „Artificial societies“[4] beschreiben Rosaria Conte und Nigel Gilbert die Entwicklung der sozialen Simulationen. Wurden zu Beginn noch Modelle einer sozialen Gemeinschaft und die für sie geltenden Theorien getestet, werden immer mehr die verschieden Dynamiken von Prozessen untersucht, die soziales Lernen steuern. Bei der Untersuchung der verschiedenen sozialen Strategien rückt die Begründung erfolgreicher oder wünschenswerter Strategien in den Fokus der Untersuchung. Nicht eine natürliche soziale Gemeinschaft ist Untersuchungsgegenstand, vielmehr werden ideale, künstliche Gemeinschaften betrachtet. Mit der Konstruktion neuer sozialer Welten werden der Wissenschaft neue Bereiche erschlossen. Das Ziel ist nicht die Entwicklung neuer sozialer Welten um ihrer selbst Willen, sondern die Optimierung von *real life* Prozessen. Die Vorteile der Computersimulation sind

- die Entdeckung bisher unbekannter Effekte und
- die Entstehung von Alternativen zu bisher üblichen Verfahren.

Die Funktion gegebener sozialer Phänomene kann isoliert untersucht werden. Dabei werden existierende oder fiktive soziale Gemeinschaft explizit modelliert. Das in dieser Arbeit vorgestellte Krisenmodell beschreibt nicht die Realität. Es ist vielmehr ein Gedankenexperiment, in dem die Funktionsweise von Mechanismen der Eskalation modellanalytisch und isoliert von anderen Effekten untersucht wird. Dieses Vorgehen ermöglicht

[3] Klaus–Jürgen Langer, [76], 1989, Seite 3.
[4] Rosaria Conte, [34], 1995.

eine relativ exakte Prüfung komplizierter Situationen, ohne all die Aspekte mitberücksichtigen zu müssen, die letztlich nicht mehr erkennen lassen, ob das Zusammenwirken verschiedener situativer Faktoren die allgemeinen Gesetzmäßigkeiten übertönen. Die vorliegende Krisensimulation ist nur ein erster Schritt zur Darstellung von Krisen in einer bestimmten Simulationsumgebung. Sie beschränkt sich auf Handlungen und Handlungstypen in einem eskalierenden System. Da die Handlungen aus einer politikwissenschaftlichen Darstellung von Krisen extrahiert werden, ist die Anzahl deeskalierender Faktoren gering. Jedoch ist schon aus dieser einfachen Simulation mit Handlungstypen ersichtlich, daß eine Krisensimulation mit einfachen Mitteln möglich und ergiebig ist. Im Gegensatz

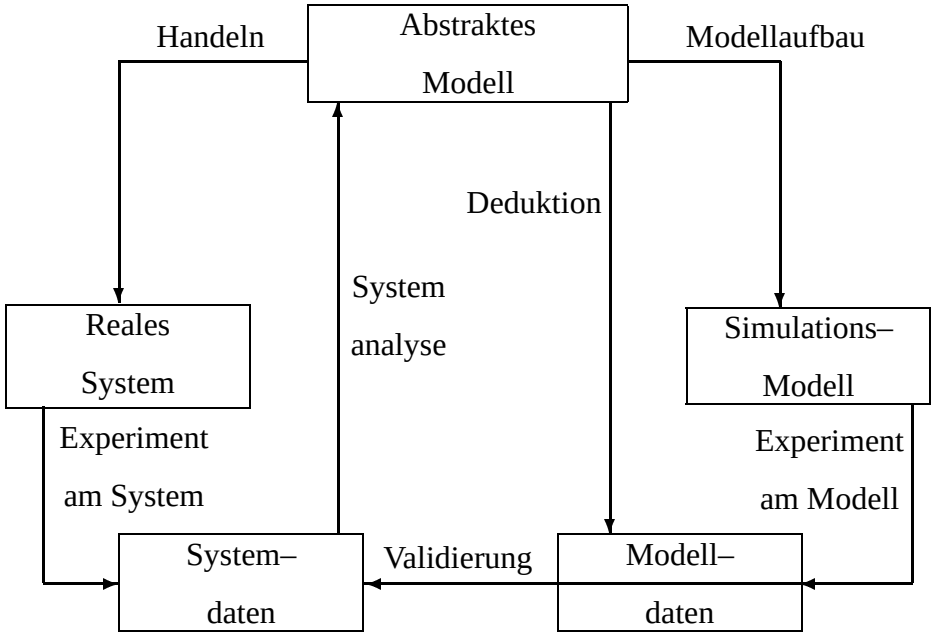

Abbildung 1: Wissenschaftlicher Erkenntnisprozeß,

nach Klaus–Jürgen Langer, [76], 1989, Seite 2.

zu einem rein militärischen Ansatz[5], ist der hier gezeigte Weg wesentlich globaler und in alle Richtungen offen. Einerseits ist eine Spezialisierung möglich, andererseits kann das

[5]zum militärischen Ansatz siehe Abschnitt 2.2 ab Seite 19.

Krisenprogramm leicht erweitert werden.

Für Simulationen ist die Beschreibung einer Krise aus der Sicht einzelner Akteure[6] am sinnvollsten. Durch dieses Vorgehen kann gezeigt werden, daß verhältnismäßig einfache Regeln für Akteure komplexe Sachverhalte hervorbringen[7]. Diese komplexen Sachverhalte sind Simulationsergebnisse, die anschließend zur Modellbildung dienen[8].

In verschiedenen Wissenschaftszweigen sind schon Versuche der Krisensimulation unternommen worden[9]. Meist verwenden diese Simulationen schon vorhandene Datenbanken[10]. Die Computersimulationen untersuchen die Datenbanken, um das Verhalten der beteiligten Akteure zu erzeugen. Um möglichst genaue Ergebnisse zu erzielen, werden die Datenbasen und Regeln erweitert, bis auch moderne Computer überfordert sind. Die Komplexität einer Krisensimulation veranschaulicht die Befragung des Generals Paul Gorman durch Senator Strom Thurmound 1992 anläßlich eines Hearings über Simulationstechnologie:

> „ … I would imagine that if you could model political and economic factors, you could make simulators for crisis management training as well. Have you recommended any analysis in this area?“[11]

Die Antwort des Generals ist ziemlich pessimistisch:

> „ … My experience makes [me] a strong supporter of simulations as a way of prepairing for future crisis. That same experience leaves me douptful that we will ever be abel to adequately model political and economic factors. … Modeling war is comparatively simple.“[12]

Jede Computersimulation muß das technisch Machbare mit dem kognitiv Plausiblem kombinieren. Der hier gezeigte Weg schlägt eine neue Richtung ein. Aus abstrakten

[6] Macchiavelli (†1527) beschreibt als erster eine politische Handlungslehre aus Sicht eines Akteurs. Siehe dazu Frank Pfetsch, [100], 1995.

[7] siehe dazu auch Robert Axelrod, [1], 1997.

[8] Zur Modellbildung außerhalb der Physik siehe Wolfgang Stegmüller, [117], 1986, ab Seite 360.

[9] siehe dazu Abschnitt 2.2 auf Seite 20.

[10] siehe dazu Abschnitt 2.3 auf Seite 22.

[11] Office of Technology Assessment, [93], 1994, Seite 33.

[12] Office of Technology Assessment, [93], 1994, Seite 33.

Handlungstypen wird eine Krisensimulation entwickelt.

Ziel der vorliegenden Arbeit ist der Nachweis, daß sich Krisen auf der Basis von relativ kleinen strukturierten Systemen von Handlungstypen modellieren und simulieren lassen. In den bisherigen Krisensimulationen[13] werden Regeln und einschlägige Begriffe aus Datenbanken abstrahiert. Die Regeln und Begriffe sind in den Datenbanken schon enthalten. Sie sind daher kontingente Bestandteile und Begriffe der behandelten Krisen und nur für eine oder wenige konkrete Krisen relevant. In dieser Arbeit wird dagegen ein abstraktes strukturiertes System von H-Typen benutzt, das aus der politwissenschaftlichen Forschung stammt. Im Gegensatz zu den maschinell gefundenen „operationalen " Regeln der existierenden Simulationssysteme werden hier mehr theoretisch strukturierte Begriffe und Regeln verwendet. Mit dem Aufbau eines derartigen Simulationsprogramms soll gezeigt werden, daß die Methode der handlungstypgestützten Simulation auch für die Politwissenschaft, speziell für die Krisenforschung ein einfaches, aber mächtiges Werkzeug darstellt. In diesem Sinn leistet die Arbeit einen Beitrag zur Methodologie der Sozialwissenschaften. Das Ziel ist nicht, eine möglichst wirklichkeitsnahe Simulation zu erzeugen, sondern neue Wege zu Gesetzmäßigkeiten und Modellen von Krisen zu finden.

Gerade bei nichtphysikalischen Theorien müssen Vereinfachungen und Idealisierungen in Kauf genommen werden[14], bis der Präzisierungsgrad der Theorie hoch genug für weitere Konkretisierungen ist. Die Simulation soll das Wissen über Krisen diesem Ziel ein Stück näher bringen.

> „ ... the AI/IR[15] literature has most of the characteristics of a classical Kuhnian paradigm, including a new set of questions, theories of data, and techniques. As a paradigm, AI/IR allows one to look at old data in new ways; ... ."[16]

[13] siehe dazu Abschnitt 2.3 auf Seite 22.

[14] siehe Wolfgang Stegmüller, [117], 1986, Seite 361.

[15] AI/IR **A***rtificial* **I***ntelligence of* **i***nternational* **r***elations.*

[16] Philip Schrodt, [114], 1991, Seite 26.

2 Krisen

Konflikte sind nicht nur universell, sondern auch normal und notwendig.

> „Conflict is not only universal but also normal and necessary in the sense that every single person and every group has its own needs, expectations, and ways of behaving that is regards as appropiate. Given this diversity, and given that we live in a world of limited resources and opportunities, it is not surprising that conflict is a normal part of life.“[17]

Drohung von
Einsatz der Armee
Blockade
Landbesetzung
Kriegserklärung
Zurschaustellung militärischer Stärke
militärische Alarmbereitschaft
Mobilisierung
Militärische Mittel
einzelne Scharmützel
Blockade
Besetzung eines Territoriums
Beschießung begrenzter Ziele für kurze Zeit

Tabelle 1: Handlungen in einer ernsten Krise
nach Frank Pfetsch, [98], 1990, Seite 12.

Die Auswirkungen internationaler Konflikte werden zunehmend begrenzt. In der UNO wirken die Großmächte nach Beendigung des „Kalten Krieges“ zunehmend konfliktthemmend. Die politischen Regionalorganisationen wie KSZE/OSZE, OAU, OAS, ASEAN, AL, IC besitzen eigene Instrumente des Konfliktmanagements. Die Wirtschaftskooperationen wie GATT, WTO, OECD, EU, NAFTA, APEC, MERCOSUR tragen ebenso zu einer Eindämmung von Streitfällen bei[18]. Trotzdem ist die Anzahl der mit Waffen ausgetragenen Konflikte hoch[19]. Seit 1945 starben zwischen 15 und 30 Millionen Menschen[20] an den Folgen kriegerischer Handlungen.

[17] Jacob Bercovitch, [17], 1997, Seite 1
[18] siehe dazu Frank R. Pfetsch, [101], 1996.
[19] siehe dazu auch Abschnitt 2.1 auf Seite 10
[20] Jacob Bercovitch, [17], 1997, Seite 3. Diese Zahl bezieht sich auf die direkten Opfer von Kampfhandlungen addiert mit den Opfern von Hungersnöten und Seuchen als Folgen von Kriegen.

2.1 Krisendefinition

Organisationen oder Individuen werden von Zeit zu Zeit extremen Situationen ausgesetzt – den Krisen. Krisen zeichnen sich aus durch Zeitbegrenzung, wenig oder ungenügende Information und dadurch, daß eine Fehlentscheidung schwerwiegende Folgen haben kann[21]. Nationale Krisen können dramatische Ausmaße annehmen. Nicht nur Individuen oder kleine Gruppen sind durch Armut, Elend, Hunger, Arbeitslosigkeit, Verfolgung, Folter, Krieg oder Tod bedroht[22]. Krisen können völlig unerwartet auftreten und mit bewaffneter Konfrontation oder sogar Krieg verbunden sein. Die Entscheidungsträger in diesen Krisen sind meist hochrangige Politiker, die an diplomatische Protokolle und Techniken gebunden sind und durch verschiedene Institutionen unterstützt und kontrolliert werden. Die Zeiten des „Kalten Krieges“ sind vorbei. Stets war jedoch eine „Superkrise“ mit der Möglichkeit einer Eskalation in einen Nuklearkrieg unwahrscheinlich. Eine Superkrise ist natürlich spektakulär, da von ihr die gesamte Weltbevölkerung betroffen ist. Aber die Krisen, in die Kräfte am unteren Ende der Gewaltskala (Terrorismus, Umsturz, regionale Konfrontationen) involviert sind, haben ihren festen Platz in der Tagesordnung[23]. Eskalierende Krisen haben teure militärische Interventionen oder Weltkriege zur Folge. Wenn die Eskalationskette nicht unterbrochen wird, sind die Konsequenzen meist disaströs.

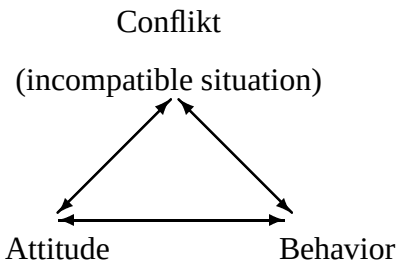

Abbildung 2: Konfliktelemente,
nach Johan Galtung, [50], Seite 125.

[21] siehe Robert H. Kupperman, [75], 1987, Seite 1.
[22] siehe Manuel Flury, [45], 1983, Seite 3.
[23] Die Datenbank KOSIMO, [102], verzeichnet 1990 182 Staaten, 1057 Konfliktbeteiligungen von 138 Staaten. In einer separaten Putschliste werden 367 Putschs und Putschversuche geführt.

Der Begriff des „Konflikts“ läßt sich gegen den Begriff des „Spiels“ oder „Wettbewerbs“ (ökonomische Konkurrenz) abgrenzen[24]. Das Spiel erfolgt nach festen Regeln, die von einem Schiedsrichter überwacht werden. Nur eine Mannschaft kann gewinnen. Das Ziel wirtschaftlicher Wettbewerbe sind bestimmte Güter. Die Wünsche der Kontrahenten bestehen aus unvereinbaren Gegensätzen. Kommt es zur Verletzung der relativ losen Regeln, kann ein Gericht angerufen werden. Bei politischen Konflikten fehlen sowohl die Regeln als auch die Gerichtsinstanz. Zwar gibt es internationale Gerichte, denen aber meist die Exekutivorgane fehlen. Sind Exekutivorgane vorhanden operieren sie nahezu wirkungslos. Die Interessensgegensätze der Staaten beziehen sich auf nationale Werte wie territoriale Unabhängigkeit, nationale Selbstbestimmung und das Entscheidungsmonopol des Staates. Da diese Merkmale einen Staat kennzeichnen, muß auf der einen Seite eines Konflikts immer die organisierte Staatsmacht beteiligt sein.
Pfetsch unterscheidet fünf Konfliktkategorien[25]:

1) **latenter Konflikt:** Interessensgegensätze werden im Unterschied zum *ruhenden Konflikt* geäußert. Die Äußerungen sind unterhalb des gewaltsamen Mitteleinsatzes. Latente Konflikte dauern durchschnittlich sieben Jahre.

2) **Krise:** Eine Krise führt zur Intensivierung des Spannungszustandes im Vorfeld militärischer Drohungen. Die Wahrscheinlichkeit militärischer Auseinandersetzungen ist erhöht. Ein Beispiel für die Intensivierung sind wirtschaftliche Sanktionen. Krisen haben eine durchschnittliche Dauer von drei Jahren.

3) **ernste Krise:** In diesem Stadium von Krisen wird mit Gewalt gedroht oder tatsächlich Gewalt eingesetzt.

4) **gewaltsamer Konflikt:** Ein gewaltsamer Konflikt ist eine Steigerung der ernsten Krise mit militärischem Mitteleinsatz. Ein gewaltsamer Konflikt, der im Durchschnitt fünf Jahre dauert, zeichnet sich durch drei Komponenten aus:

[24] siehe Frank R. Pfetsch, [98], Seite 9.
[25] siehe Frank R. Pfetsch, [98], Seite 11.

- Am Kampf sind zwei oder mehr bewaffnete Streitkräfte beteiligt, wobei auf mindestens einer Seite reguläre Regierungsgruppen stehen.
- Es existiert ein Mindestmaß zentraler Organisation auf beiden Seiten.
  Durch die Kampfhandlungen sind Opfer zu beklagen.
- Die Kampfhandlungen folgen einer planmäßigen Strategie.

5) **Kriege:** Kriege treten nur zwischen etwa gleichstarken Gegnern auf. Sie sind von einiger Dauer, fordern Opfer und bringen Zerstörung. Kriege lassen sich unterscheiden nach:

   - Angriffs– und Verteidigungskrieg,
   - gerechter und ungerechter Krieg,
   - kolonialer und imperialer Krieg,
   - konventioneller und nuklearer Krieg,
   - totaler und begrenzter Krieg.

Die Intensität der Konflikte ist abhängig von der Anzahl involvierter Staaten, externen Interventionen und ideologischer Aufladung der Konflikte. Sind diese drei Elemente ausgeprägt, steigern sie die Komplexität und wirken konfliktverschärfend. Nur in 20% der Fälle ist ein Aggressor auch militärisch erfolgreich. Werden Gewaltmittel eingesetzt, kann ein Konflikt durch Niederlage, Sieg oder Waffenstillstand (Patt) beendet werden.
Das grundlegende Problem der Krisendefinition ist Krisen zu identifizieren. Was unterscheidet eine Krise vom normalen Status?
Als Voraussetzung für Konflikte identifizieren Mack und Snyder[26] folgende Punkte:

- die Existenz von zwei oder mehr Parteien,
- die Interaktionen der Parteien entwickeln sich aus einer Knappheit von Ressourcen oder einer Unvereinbarkeit ihrer Positionen,

[26] R. W. Mack, [80], 1957

- die Akteure verwickeln sich in gegeneinander gerichtete Handlungen,
- das Verhalten hat das Ziel, den anderen zu beschädigen, zu verletzen oder zu eliminieren,
- die Interaktionen sind offenkundig und können von Beobachtern gemessen und bewertet werden.

Solche Konflikte stehen im Gegensatz zur Kooperation.
Unterschiedliche Autoren bieten verschiedene Definitionen[27] einer Krise an. Jedoch sind fast allen Krisendefinitionen mehrere der folgenden Punkte gemeinsam:

- Ereignis – Ausbruch[28],
- veränderter Interaktionsfluß[29],
- zerrüttete oder zerbrochene Interaktion[30],
- Systemgefährdung[31],
- Systemveränderung[32],
- Bedrohung[33]
- Kriegsgefahr[34],
- Zeitknappheit[35],

[27] Zur Definition von Krisen siehe auch Herman Kahn, [69], 1965 und Heinz Krummenacher, [74], Seite 19 – 40.
[28] siehe Charles McClelland, [84], 1968, Seite 161.
[29] siehe Charles McClelland, [84], 1968, Seite 160.
[30] siehe Michael Brecher, [25], 1988, Seite 3.
[31] siehe Charles Hermann, [64], 1972, Seite 10, sowie Michael Brecher, [25], 1988, Seite 3.
[32] siehe Michael Brecher, [23], 1979, Seite 6.
[33] siehe Charles Hermann, [64], 1972, Seite 13, Michael Brecher, [23], 1979, Seite 6, Richard Lebow, [77], 1981, Seite 10, sowie Michael Brecher, [25], 1988, Seite 3.
[34] siehe Michael Brecher, [23], 1979, Seite 6, Richard Lebow, [77], 1981, Seite 10, sowie Michael Brecher, [25], 1988, Seite 3.
[35] siehe Charles Hermann, [64], 1972, Seite 13, Michael Brecher, [23], 1979, Seite 6, Richard Lebow, [77], 1981, Seite 12, sowie Michael Brecher, [25], 1988, Seite 3.

- Überraschung[36],
- Ungewißheit.

Dabei werden zwei Typen von Definitionen unterschieden. Einerseits wird eine Krise auf der Systemebene betrachtet, andererseits wird die Krise auf der Ebene der Entscheidungsträger definiert. Manche Autoren wie zum Beispiel Hermann[37] und Brecher[38] führen dazu eine entsprechende doppelte Definition an. Obwohl mehrere Autoren denselben Begriff für eine Krisendefinition verwenden, unterscheiden sich ihre Definitionen doch erheblich. Zum Beispiel bedeutet „Bedrohung“

- bei Brecher eine „Bedrohung der Ziele“,
- bei Hermann eine „Bedrohung der Werte“ und
- bei Lebow eine „Bedrohung der Werte und Positionen“.

Nach Manuel Flury[39] werden Krisen durch sechs Kriterien bestimmt:

1) Eine Krise ist ein Wendepunkt der betroffenen Gesellschaft.
2) Krisen bedrohen Ziele und Absichten der Beteiligten.
3) Mit Krisen sind erhöhte Auseinandersetzungen zwischen den Betroffenen verbunden.
4) Krisen reduzieren die Kontrolle über Geschehnisse und Auswirkungen.
5) Krisen bedeuten Veränderungen in Gesellschaften.
6) Krisen bedeuten Funktionsstörungen in verschiedenen Sektoren der Gesellschaft.

[36] siehe Charles Hermann, [64], 1972, Seite 13.
[37] siehe Charles Hermann, [64], 1972, Seite 9–13.
[38] siehe Michael Brecher, [25], 1988, Seite 9 – 11.
[39] siehe Manuel Flury, [45], Seite 15.

Diese Krisendefinition ist auf Flurys Konzept sozialer Konflikte und Spannungen zugeschnitten. Dementsprechend sucht Flury auch die Ursachen von Krisen mehr im ökologischen, demographischen, sozioökonomischen und ethnisch–kulturellen Bereich. Seine Krisendefinition betrifft vor allem innerstaatliche Konflikte.
Im „Handbook of International Crisis“ unterscheidet Brecher[40] eine Krisendefinition auf dem Mikrolevel und Makrolevel. Die Definition auf dem Mikrolevel basiert auf der Sicht der einzelnen Staaten (*crisis viewed from the perspective of an individual state*). Eine Krise zeichnet sich aus durch eine:

- Bedrohung von Grundwerten,
- Fehlen von Zeit zur Wahl einer Antwort und
- große Gefahr von Verwicklung in militärische Auseinandersetzungen.

Die Definition auf dem Makrolevel basiert auf objektiven Daten über Konfliktinteraktionen. Die drei Bestandteile dieser Krisendefinition sind:

- zerrüttende oder zerbrechende Interaktion zwischen einem oder mehreren Gegnern,
- eine hohe Kriegsgefahr oder falls der Krieg schon ausgebrochen ist, eine Unausgewogenheit der militärischen Balance,
- eine existierende Gefahr für das bestehende System.

Brechers Krisendefinition in „Toward a Theorie of International Crisis Behavior“ unterscheidet zwischen konzeptueller und operationaler Krisendefinition[41].
Brechers konzeptuelle Krisendefinition lautet:

> „ . . . a foreign policy crisis is a situational change in the external or internal enviroment which creates in the minds of the incumbent decision-makers of an international actor a perceived threat from the external enviroment to [the] basic values to which a responsive decision is deemed necessary.“[42]

[40] Michael Brecher, [25] Band I und [26] Band II, 1988.
[41] siehe Michael Brecher, [22], 1977, Seite 42.
[42] Michael Brecher, [22], 1977, Seite 43.

Die operationale Krisendefinition von Brecher ist:

> „ ... a foreign policy crisis is a breakpoint along the peace-war continuum of a state's relations with any other international actor(s). A crisis is a situation with four necessary and sufficient conditions, as these are *perceived* by the highest level decision-makers of the actor concerned:
>
> 1) a change in its external or internal enviroment, which generates
> 2) a threat to basic values, with a simultaneous or subsequent
> 3) high probability of involvement in mititary hostilities, and the awareness of
> 4) a finite time for their response to the external value threat."[43]

Die letzte Definition nimmt Rücksicht auf vier essentielle Punkte einer Krise:

1) Überraschung,
2) hohe Wahrscheinlichkeit von militärischen Auseinandersetzungen,
3) endliche (kurze) Zeit für die Antwort und
4) eine Änderung der Situation liegt im Möglichkeitsbereich des Akteurs.

Ein Bestandteil dieser Definition ist, daß die Krise nicht direkt durch eine Handlung des Akteurs ausgelöst wird. Obwohl das krisenauslösende Moment nicht im Handlungsbereich der Agenten liegt, ist es sehr wohl möglich, daß die Krise als eine Reaktion auf eine Aktion eines Beteiligten eintritt.

[43] Michael Brecher, [22], 1977, Seite 43.

Ergebnis	Anzahl	Prozent
Abzug Militär	22	8.1
Krieg geht weiter	34	12.5
Niederlage des Initiators	83	30.4
Sieg des Initiators	50	18.3
Waffenstillstand	84	30.8

Tabelle 2: militärische Ergebnisse

nach Frank Pfetsch, [103], 1994, Seite 184.

2.2 Künstliche Intelligenz und internationale Politik

Simulationen zum Krisenmanagement haben eine lange Tradition. Die Wurzeln liegen in schachähnlichen Kriegsspielen des fünfzehnten Jahrhunderts[44]. Die ersten echten Kriegsspiele wurden 1811 von Reisswitz, einem preußischen Leutnant, entwickelt. Sein Spiel besteht aus einer Karte und Spielsteinen, die Truppen zu verkörpern. Zwei Spieler werden von einem Schiedsrichter überwacht. Dazu existiert ein Buch mit vielen detaillierten Regeln[45].

Reale Simulation[46] der RAND Cooperation in den fünfziger und des MIT Centers für internationale Studien in den sechziger, sowie CSIS in den achtziger Jahren waren aufwendig und teuer.

In den letzten Jahren werden verstärkt neue Simulationen und Spiele auf dem Computer oder dem Spielbrett genutzt. Der Einsatz ist vielfältig. Politische, militärische, medizinische, ökonomische und pädagogische Zwecke werden verfolgt. Ein weitere, oft verfolgte, aber hier nicht untersuchte Absicht der Simulationsspiele ist auch das persönliche Vergnügen. Einen breiten Raum nehmen hier auch Kriegsspiele ein[47]. Die Simulation und das Ersetzen verschiedener Aspekte der Realität dienen vier verschiedenen Zielen:

- dem Experimentieren,
- der Vorhersage,
- der Bewertung oder
- dem Lernen[48].

[44] Zu einer Geschichte des Kriegspiels siehe Henk A. Becker, [14], 1980.

[45] siehe J. Lewis Rasmussen, [108], 1992, Seite 3.

[46] Der Begriff „reale Simulation“ bedeutet die Simulation mit Personen in einer Umwelt, die möglichst realitätsnah gestaltet ist. Während der Simulationsdauer werden die Personen von ihrer normalen Umwelt isoliert.

[47] In „The Electronic Battlefield“ von Bob Guerra, [61], werden bereits 1987 dreißig ausgewählte Computerkriegsspiele vorgestellt. Eine dieser Simulationen der Avalon Hill Game Company heißt *Gulf Strike* (Seite 187 – 194). Diese Simulation erschien 1986. Dazu der Autor: „This Simulation, though not an historical renactment of recent events, gives a sense of urgency because of its closeness to the contemporary world.“ (Seite 187). Der Golfkrieg fand erst 1994 statt.

[48] siehe Barton J. Cunningham, [36], Seite 215.

Der einfache und kostengünstige Einsatz von Computersimulationen ermöglicht es, viele Simulationen umfangreicher und öfter durchzuführen als dies bei realen Simulationen[49] möglich wäre. Ist ein Programm erst einmal erstellt, können leicht einige Parameter geändert werden.

Natürlich sind Daten realer Simulationen notwendig. Aber diese Daten können am Computer leicht so modifiziert werden, daß sie ein optimales[50] Ergebnis erreichen. Noch befindet sich die Computersimulation von Krisen in den Anfängen, da bei einer Krise viele Faktoren berücksichtigt werden müssen. Krisen sind so vielfältigen Struktur–, Verhaltens– und Interdepenzvariablen ausgesetzt, daß Einschätzungen schwierig sind. Aber schon jetzt beeinflussen die Ergebnisse dieser Simulationen das Wissen über Krisen. Auch die militärischen Computersimulationen sind ein Bereich der Krisensimulationen. Sie reichen von der aufwendigen Gefechtssimulation[51], über Flugsimulatoren[52] bis hin zur Berechnung von Raketenflugbahnen. Für jeden Bereich existieren aufwendige und teilweise sehr spezielle Simulationen[53]. Der Bereich der Computerdarstellungen reicht vom U-Boot[54] bis zum Satelliten. Schon 1943 wurde einer der ersten Computer ENIAC[55] zur Berechnung von Wasserstoffbombenexplosionen eingesetzt.

Die meisten modernen Simulationssysteme basieren auf der virtuellen Realität. Dabei agieren Personen über Sensoren und Effektoren mit einer synthetischen Umwelt als ob diese real wäre. Bekannte Elemente sind ein Datenhelm, der die Ton– und Bildausgabe der Simulation übernimmt, sowie ein Datenhandschuh, mit dem Eingaben auf einer virtuellen Konsole gemacht werden können. Hardwareaufwendige Cockpitsimulationen werden allmählich durch diese beiden leichter einsetzbaren Elemente ersetzt.

[49] siehe Fußnote 46 aus Seite 18.

[50] Optimale Ergebnisse können besonders realitätsnah, aussagekräftig usw. sein.

[51] siehe dazu Joseph J. Molitoris, [89] 1995.

[52] siehe z.B. A. M. Cameron, [28], 1995 und Ki Cheon Yoon, [126], 1995.

[53] Ein ausführliches Literaturverzeichnis zur virtuellen Realität und den Technologien zur Gefechtssimulation findet sich in einem Hintergrundbericht des Office of Technology Assessment, [93], 1994.

[54] siehe John M. Brett, [27], 1995.

[55] ENIAC – Electronic Numerical Integrator and Calculator.

Bei der Entwicklung der oben genannten Systeme wurden Krisensimulationen, die den militärischen Einsatz verhindern sollen, vernachlässigt. Da ein krisenbasierter Ansatz wesentlich umfangreicher als eine Gefechtssimulation ist, wird es natürlich schwieriger, alle Bestandteile der realen Welt nachzubilden. Sowohl politische Entscheidungen als auch militärisch Machbares sollte in solche Simulationen einfließen. Daß dieser Ansatz äußerst komplex ist, wurde bereits in Abschnitt 1 auf Seite 7 ausführlich dargelegt. Da aber Roboter[56] bereits ins Kinderzimmer vordringen, ist anzunehmen, daß auch auf diesem Sektor erhebliche Fortschritte gemacht werden.

> „Daneben haben auch Fortschritte in der Datenerhebung und –verarbeitung und die Anwendung neuer statistischer Analysetechniken die empirische Forschung innerer Unruhen und Kriege auf breiter Basis gefördert (vor allem Studien mit cross-nationalen Aggregatdaten).“[57]

Valerie Hudson beschreibt in ihrem Buch „Artificial Intelligence and International Politics“[58] die Entwicklung sozialer Simulationen im Bereich der politischen Wissenschaften und sucht nach Erklärungen für internationales Verhalten relevanter Personengruppen, wie Staatsführer, Außenpolitiker und Politikwissenschaftler. Computersimulationen von internationalen Relationen sind ein kleiner Bereich der *Artificial Intelligence*. Für Krisensimulationen sind die Grundansätze der AI meist nicht sehr produktiv:

> „You can’t model politics using artificial intelligence; you’d have to use artificial stupidity.“[59]

Ein Krisenszenario zeichnet sich gerade durch mangelnde Information, mangelnde Rationalität, Gruppenentscheidungsprozesse und irrationale Entscheidungen aus. Politische Maßnahmen sind meist nicht rational. Rationale Entscheidungen setzen voraus, daß die Konsequenzen des Handelns bekannt sind. Aber internationale Konflikte sind so komplex, daß ein Wissen über Handlungsfolgen unrealistisch erscheint. Die AI wird also

[56] gemeint ist das LEGO Mindstorms, Robotics Invention System. Siehe dazu Andrew Baum, [12], erscheint im April 2000, sowie Dave Baum, [13], 1999, sowie Jonathan B. Knudsen, [71], 1999.

[57] Beat Moser, [90], 1974, Seite 1.

[58] Valerie M. Hudson, [66], 1991.

[59] Philip Schrodt, [114], 1991, Seite 10.

verwendet, um die Simulationswerkzeuge zu verbessern und nicht die in der Simulation verwendeten Prozesse. Auch in KRIS wird genau dieser Ansatz verfolgt. Die Handlungsentscheidungen sind simplen Regeln oder dem Zufall unterworfen. Die meisten AI/IR–Systeme[60] kodieren ihr Wissen in der Form von *if ... then*–Regeln[61]. Ein regelbasiertes System spiegelt politisches Verhalten ungewöhnlich gut wieder, da politisches Handeln und Verwaltungsakte explizit auf festen Regeln aufbauen. Gesetze und Vorschriften sind nichts anderes als Regeln. Diese Richtlinien können unscharf formuliert sein und sie determinieren sicher nicht das Verhalten, aber sie beeinflussen es weitgehend. Informelle Regeln beeinflussen die Aktionen der Entscheidungsträger[62]. Das regelkonforme, politische Verhalten kann auch empirisch nachgewiesen werden. Ein regelbasiertes System besteht aus drei Teilen:

1) einer Menge von Regeln,
2) einer Datenbasis und
3) einer Möglichkeit, die Regeln auszuführen, um eine Aufgabe zu erfüllen oder ein Problem zu lösen[63].

[60] AI/IR **A***rtificial* **I***ntelligence of* **i***nternational* **r***elations.*

[61] Philip Schrodt, [114], 1991, Seite 17.

[62] Ein Beispiel ist, daß John F. Kennedy während der Kubakrise nicht in Betracht zog, den sowjetischen Botschafter und seine Familie als Geiseln zu nehmen. Dieses Verhalten war ca. 200 v. Chr. im Römischen und Persischen Reich üblich.

[63] siehe Dwain Mefford, [86], 1991, Seite 61.

2.3 Datenbanken

Es existieren vier große Datenbanken über internationales Konfliktmanagement. SHERFACS[64] wurde von Frank L. Sherman ins Leben gerufen. Im Unterschied zu den anderen vorgestellten Datenbanken, werden in SHERFACS die Daten statt in Tabellenform mit einer Baumstruktur erfaßt. Dabei verwendet SHERFACS fünf verschiedene Datenebenen:

1) Fallübersicht,

2) Entwicklungsstruktur,

3) Handlungen der Beteiligten,

4) Management durch Unbeteiligte und

5) Management der Handlungen der Beteiligten.

Jacob Bercovitch erstellte die Datenbank CONFMAN[65] zu diesem Thema. Er legt besonderen Wert auf die Vermittlung dritter Mächte bei Lösungen von Konflikten. Er untersucht 241 internationale Krisen zwischen 1945 und 1990[66]. In 60% der Fälle waren Vermittler beteiligt. Bei diesen Streitfällen fand Bercovitch fast 600 Vermittlungsversuche. Er kodiert die Konflikte mit 50 Variablen. Zu dieser Datenbank existiert auch ein Programm, das automatisch Regeln aus diesen Daten ableitet. Dabei werden Entscheidungsbäume generiert, die Regeln wie in Tabelle 3 gezeigt, erstellt.
Die dritte große Datenbank KOSIMO[67] ist in Deutschland entstanden. Frank Pfetsch und Peter Billing haben sie entwickelt und erweitern diese Datenbank fortlaufend. Diese Datenbanken werden immer wieder in Buchform veröffentlicht[68]. Die Version September

[64] siehe Frank L. Sherman, [116], 1999.
[65] siehe Jacob Bercovitch, [16], 1999 und Jacob Bercovitch, [15], 1996, sowie Jacob Bercovitch, [17], 1997.
[66] Eine neue Version seiner Konfliktdatenbank untersucht 268 Konflikte.
[67] siehe Frank Pfetsch, [102], 1999.
[68] siehe Frank R. Pfetsch, [99], 1991 und Frank R. Pfetsch, [103], 1994, sowie Frank R. Pfetsch, [101], 1996.

Wenn	es weniger als 400 Opfer	**und**
	Akteur *B*'s Stärke ist ≤ 33	**und**
	das Konfliktmanagement basierte auf Vermittlung	**und**
	der Konflikt dauerte zwischen einem und drei Monaten	
dann	war die Vermittlung immer erfolgreich	
	in 15 Vermittlungen bei 8 verschiedenen Konflikten	

Tabelle 3: Regel extrahiert aus Confman,

nach Johannes Fürnkranz, [48], 1994, Seite 11.

1999 von KOSIMO enthält 661 interne und internationale Konflikte. Jeder dieser Konflikte ist mit dreißig Variablen[69] codiert. Zu KOSIMO existiert ein CBR (**C***ase* **B***ased* **R***easoning*) Programm zur Auswertung[70]. Dabei werden aus den Daten Regeln abstrahiert, um die Konflikte zu klassifizieren und zu erklären. Auch Michael Brecher hat in seinem *Handbook of International Crisis*[71] eine Liste von Krisen veröffentlicht.

[69] siehe Tabelle 18 auf Seite XXII.
[70] siehe Johann Petrak, [96], 1994.
[71] siehe Michael Brecher, [25] und [26], 1988.

3 Handlungen

Handlungen haben eine Entsprechung in der Sprache – „die Verben“. Zur Gruppierung von Verben kann der Ansatz von Thomas Ballmer wertvolle Hinweise leisten.
Auch in klassischen Handlungstheorien wird versucht die Fülle von möglichen Handlungen zu sortieren. In Abschnitt 3.2 werden einige Möglichkeiten aufgelistet.

3.1 Verben

In der deutschen Sprache gibt es nach Mater[72] ca. 20.000 Verben. Eine Einteilung von Verben nach Zustands–, Vorgangs–, und Tätigkeitsverben ist ungenügend[73]. Ein strukturierter Verbwortschatz reflektiert sozial relevante Handlungs– und Vorgangsmuster[74]. Zum Verbkomplex gehören einerseits das Verb, andererseits die nominalen Satzglieder und adverbialen Bestimmungen. Das Verb wiederum zerfällt in Verbstamm und Verbmorphem[75]. Die Verbstämme tragen die inhaltliche Bedeutung der Verben und sind somit für die Untersuchung logischer Relationen[76] zwischen Verben geeignet. Die Struk-

Strukturierungsebene	Anzahl Einheiten
Kategorie VERB	1
Verbtypen	3
Modellgruppen	11
Modelle	40
Kategorien	700
einfache Verben	8.000
gebräuchliche Verben	13.000
Verben	20.000

Tabelle 4: Strukturierungsebenen deutscher Verben, nach Thomas Ballmer, [4], 1986, Seite 11.

[72] siehe Erich Mater, [81], 1966.
[73] siehe Thomas Ballmer, [4], 1986, Seite 3.
[74] siehe Thomas Ballmer, [4], 1986, Seite 4.
[75] Verbmorphem beinhaltet so unterschiedliche Strukturen wie Präfixe, Suffixe, Numerus, Modus und andere.
[76] Als Relationen bieten sich an: Folgerungsrelation, Präsuppositionsrelation, usw.

tur eines Wortschatzes läßt sich in ordnenden Relationen wie Hyponymie[77], Autonymie, Komplementarität darstellen oder durch Angabe von Klassen und Worttupeln, die in Relation zueinander stehen. Eine ordnende Relation liefert eine Liste abstrakter Eigenschaften, während das Bilden von Klassen eine Liste von zusammengefaßten Wörtern oder deren Tupel ergibt. Die Strukturierungsebenen der deutschen Verben nach Thomas Ballmer sind in Tabelle 4 auf Seite 24 dargestellt.

sich fortbewegen jd 1 zu Fuß von ort 3 nach ort 3	extrem langsam[78]
sich fortbewegen jd 1 zu Fuß von ort 3 nach ort 3	langsam
sich fortbewegen jd 1 zu Fuß von ort 3 nach ort 3	
sich fortbewegen jd 1 zu Fuß von ort 3 nach ort 3	gemütlich
sich fortbewegen jd 1 zu Fuß von ort 3 nach ort 3	tüchtig vorankommend
sich fortbewegen jd 1 zu Fuß von ort 3 nach ort 3	schnell
sich fortbewegen jd 1 zu Fuß von ort 3 nach ort 3	maximal schnell

Tabelle 5: Verbgruppe eingeordnet in Verbklassen,

nach Thomas Ballmer, [4], 1986, Seite 34.

Diese Einteilung verwendet eine Prototypizität, die Verben in ihrer normalen, standardmäßigen Verwendung berücksichtigt. Da nach strukturalistischer Auffassung die Sprache ein einheitliches Gefüge darstellt, folgt aus der Einführung neuer Verben meist das Einführen neuer Merkmallisten. Deshalb sollten die einzelnen Einträge in diesem Lexikon nicht überbewertet werden. Die Verben werden als ungeordnete Liste vorgegeben und die Kategorien aus den Verben entwickelt. Neben dem verwendeten Handlungsmodell der Verben gibt es noch sehr umfangreiche Lebens–, Fortbewegungs–, Greif– und Habenmo-

[77] Hyponymie bezeichnet untergeordnete Begriffe wie zum Beispiel ‚Rosen‘ und ‚Tulpen‘ bei ‚Blumen‘.

[78] Legende: jd 1 jemand im Nominativ
ort 3 Ortsangabe im Dativ.

delle. Dabei spielen die Kontextbezüge der Verben eine wichtige Rolle. Die Kontextkoordinaten (Raum, Zeit, Umwelt, Sprecher, Hörer) können nicht isoliert betrachtet werden, da sie interagieren.

Die inhaltliche Seite eines Satzes wird extensional als Wahrheitswert und intensional als Proposition aufgefaßt. Verben werden von Ballmer[79] nicht als Relation zwischen Individuen und Individuenbegriffen aufgefaßt, sondern typischerweise[80] als Prozesse, in die Individuen, Propositionen und Tatbestände involviert sind.

Unter diesen Prozeßbegriff fallen Vorgänge, Ereignisse, Geschehnisse, Tätigkeiten, Handlungen sowie Zustände, die sich über eine Zeitdauer erstrecken. Prozesse sind also materielle, geistige oder soziale Abläufe in der Zeit.

schleichen	jd 1 von ort 3 nach ort 3
zuckeln	jd 1 von ort 3 nach ort 3
gehen	jd 1 von ort 3 nach ort 3
trotten	jd 1 von ort 3 nach ort 3
marschieren	jd 1 von ort 3 nach ort 3
rennen	jd 1 von ort 3 nach ort 3
sprinten	jd 1 von ort 3 nach ort 3

Tabelle 6: Beispiel für eine Verbgruppe
nach Thomas Ballmer, [4], 1986, Seite 34.

> „Zum Wesentlichen eines Prozesses wollen wir auch die Rolle zählen, die die Individuen, Gegenstände und Sachverhalte im Prozeß spielen, nicht aber spezifische Eigenschaften und Merkmale von Individuen, die die Individuen als solche charakterisieren, und noch viel weniger wollen wir spezifische Individuen zum Wesentlichen eines Prozesses zählen. Dieser Auffassung gemäß können also Individuen verschiedener Eigenschaften, insbesondere jeweils verschiedene Individuen am selben Prozeß teilnehmen.“[81]

Ballmer abstrahiert Individuen und Sachverhalte sowie Art und Weise des Ablaufs (Intensität). Zeit und Ort kann jedoch auch zum Wesentlichen eines Prozesses zählen. Dann

[79] siehe Thomas Ballmer, [4], 1986, Seite 31.

[80] Eigenschaften, Relationen und Dispositionen haben keinen prozessualen Charakter. Die Existenz von Verben, die sich darauf beziehen, wird von Ballmer nicht geleugnet.

[81] Thomas Ballmer, [4], 1986, Seite 32.

müssen sie bei der Untersuchung berücksichtigt werden. Ballmer liefert zusammengehörende Verbgruppen, die durch Paraphrasierungstechnik einen Hinweis darauf geben, welches Satzmuster eine Verbklasse rechtfertigt. Ein Beispiel findet sich in den Tabellen 5 und 6 auf den Seiten25 und 26, die beide dieselben Verben beinhalten. Zur Klassifizierung wird einfach der größte gemeinsame Teiler (Teilsatz) aller Paraphrasen verwendet. Dabei ließ sich Ballmer von zwei Kriterien leiten:

- dem Typ von Verben[82] und
- dem Wunsch, daß die Klassen eine vernünftige Größe haben sollen.

Gewisse Verben setzen andere semantisch voraus. Solche Paare sind beispielsweise:

- „leben – sterben“ und
- „anfangen – aufhören“.

Relationen zwischen Verbklassen sind:

- Spezialisierung („bedienen – schalten“),
- zeitlich („anfangen – stattfinden – enden“),
- alternativ, sich ausschließend („passieren – ausbleiben“, oder „beabsichtigen – ausführen“)[83],
- Befähigung („riechen können – riechen“),
- Vorbereitungsprozeß („flottmachen – wegfahren“),
- Folgeprozeß („beenden – prüfen“).

Diese Beispiele belegen, daß es sprachwissenschaftlich inadäquat ist, die Bedeutungsstruktur von Verben nur in Klassen darzustellen.

[82] Der Typ entspricht einem Satzmuster, in dem Subjekt, Objekt, usw. durch Leerstellen gekennzeichnet sind.

[83] Die Relation zwischen diesen Klassen ist eine handlungstheoretische Implikation.

3.2 Handlungstheorien

Die Untersuchung der Handlungen als eine Unterordnung von Ereignissen ist ein Anliegen des Naturalismus.

> „Sie alle kommen darin überein, daß es nicht notwendig sei, einen ontologischen Unterschied zu postulieren zwischen der Art der Handlungen und der Art sonstiger Ereignisse in der Natur. Handlungen seien lediglich eine besondere Unterart der natürlichen Ereignisse.“[84]

Von v. Wright stammt die Unterscheidung zwischen Ergebnis und Folgen von Handlungen:

> „Die Ursache werde ich auch das *Ergebnis* und die Wirkungen die *Folgen* unserer Handlung nennen. Zwischen der Ursache und den Wirkungen existiert eine Bedingungs–Relation einer bestimmten Art.“[85]

Sind Handlungen konkret, nehmen sie einen einmaligen Raum–Zeit–Platz ein[86].
Die Feststellungen über Identität zweier Aussagen sind verwirrend. Meist ist diese Identität nur eine Ähnlichkeit, die ein Wiederholen oder das Zählen von Handlungen bedingt. Der Umweg über „Handlungen unter einer Beschreibung“ ist hilfreich um Handlungen zu untersuchen, ohne sie eindeutig identifizieren zu müssen[87].
Werden Handlungen durch die Ausdrücke „indem“ und „dadurch“ geordnet, können sie durch Handlungsbäume dargestellt werden[88]. So verbundene Handlungen haben eine partikuläre Identität, aber keinesfalls eine Typen–Identität.
Die Ordnungsrelation solcher Handlungsbäume heißt *by–relation*[89]. Diese Relation ist asymmetrisch, irreflexiv und transitiv. Die aufsteigenden Äste des Handlungsbaums werden abstrakter, die absteigenden konkreter. Die Handlungsbäume und die *by–relation* werden zweideutig verwendet. Einerseits dienen sie zur Ordnung von Handlungsbeschreibungen, andererseits um die Ordnung verschiedener Handlungen darzustellen.

[84] Edmund Runggaldier, [109], 1996, Seite 15.
[85] George Henrik von Wright, [125], 1984, Seite 69.
[86] siehe Edmund Runggaldier, [109], 1996, Seite 31.
[87] siehe Edmund Runggaldier, [109], 1996, Seite 43.
[88] siehe Edmund Runggaldier, [109], 1996, Seite 46.
[89] siehe Alvin Goldman, [59], 1971.

Goldmans logische Strukturanalyse bezieht sich noch nicht auf die Zeitfolge[90] von verschiedenartigen Handlungen, sondern nur auf „identische“ oder material–analytisch miteinander verknüpfte Handlungen, die gleichzeitig eintreten[91]. Durch die „Warum“–Frage steigt man im Handlungsbaum auf, durch die „Wie“ –Frage ab. Dazu hat Pfeifer[92] eine graphische Darstellung entwickelt. Die *by–relation* kann auch zum Generieren dieser Handlungen verwendet werden. Goldman nennt sie auch „GEN–relation“[93]. Eine kausal verstandene Handlungserklärung liefert Churchland:

> „Mit „*X* hat *H* getan, weil er ∅ wollte“ hat man nur dann (und offenbar sogar: genau dann) etwas Wahres gesagt, wenn gilt:
>
> 1) *X* wollte ∅; und
> 2) *X* glaubte (schätzte, sah), daß der Vollzug von *H* unter den gegebenen Umständen für ihn ein Mittel sei, um ∅ zu erreichen; und
> 3) es gab keine Handlung, von der *X* glaubte, daß er mit ihr ∅ erreichen würde, und für die er eine wenigstens gleichermaßen große Präferenz hatte wie für *H*; und
> 4) *X* hatte keinen anderen Wunsch (bzw. keine anderen Wünsche), der ihn unter den gegebenen Umständen von seinem Wunsch ∅ abgebracht hat; und
> 5) *X* wußte, wie man *H* tut; und
> 6) *X* war in der Lage, *H* zu tun.“[94]

Die Idee der *agent–causality* bezieht die Handlung auf den Handelnden selbst[95]. Handlungen können indexikalisch beschrieben werden. Der Index kann über Zeit, Ort und Person laufen[96].

> „Speziell die höheren Beschreibungen eines Handlungsbaumes setzen einen Bezug auf Handelnde voraus, insofern ihnen nicht nur allgemein intentionale Fähigkeiten zukommen, sondern auch die Fähigkeit, sich an Vergangenes zu erinnern und Zukünftiges vorwegzunehmen, im Laufe der Zeit Verschiedenes zu lernen, und nicht zuletzt die Fähigkeit, Normen und Regeln zu befolgen.“[97]

---

[90] siehe Alvin Goldman, [58], 1970.

[91] siehe dazu auch Hans Lenk, [78], 1980, Seite 138,
sowie 4.4, Seite 43.

[92] siehe Karl Pfeifer, [97], 1989, Seite 20.

[93] siehe dazu auch Carl Ginet, [57], 1990, Seite 19 und Seite 46.

[94] Paul M. Churchland, [32], 1977, Seite 312.

[95] zum Beispiel bei Roderick Chisholm, [30], 1976.

[96] siehe John Perry, [95], 1979.

[97] Edmund Runggaldier, [109], 1996, Seite 192.

Handlungen bilden das Material, aus dem soziale Systeme aufgebaut sind[98]. Der Begriff „Handlung“ muß als Grundbegriff aufgefaßt werden. Eine Handlung ist:

- wiederholbar, da sie von einem Schema erzeugt wird,
- zielgerichtet[99] und
- sprachbeladen[100].

> „Der wichtigste Punkt, ... ,ist, daß die Wahrnehmung von Handlungen einen substantiellen Teil der Sprache und ein umfangreiches Wissen über soziale Praktiken voraussetzt.“[101]

Da Handlungen für die soziale Welt so grundlegend sind, ist es sicher sinnvoll eine Krisensimulation auf Handlungen basierend aufzubauen. Wie oben gezeigt wurde, existieren viele verschiedene Möglichkeiten Ordnungen über Handlungen aufzubauen. In KRIS soll keiner Ordnung eine Vorrangstellung eingeräumt werden. Da die Ordnung von Handlungen in den Regeln implementiert wird können die verschiedenen Ordnungen untersucht werden. Weitere Möglichkeiten Handlungen zu ordnen werden in den folgenden Kapitel aufgezeigt.

[98] siehe Wolfgang Balzer, [6], 1993, Seite 89.
[99] Dieses Ziel muß nicht unbedingt verbalisierbar sein.
[100] „Sprachbeladen“ bedeutet unter anderem, daß die Sprache eine Voraussetzung ist, damit Akteure eines sozialen Modells Handlungen wahrnehmen können.
[101] Wolfgang Balzer, [6], 1993, Seite 91.

4 Handlungen und Halbordnung

Zum Ordnen von Handlungen wurde in Konstanz ein Prologprogramm entwickelt. In Abschnitt 4.2 wird dies vorgestellt. Handlungen können in einem Propositionenraum eingebettet werden. Dieser Propositionenraum bildet eine Halbordnung. Somit können auch mathematische Gesetze auf den Handlungsraum angewendet werden. Jedoch hat sich dieses Konzept für eine Simulation von Krisen als zu eng erwiesen. Trotzdem soll dieses Konzept hier kurz vorgestellt werden. Das dritte Konzept basiert auf einem Artikel von Hans Lenk. Alle drei Darstellungen verwenden Halbordnungen oder zumindest *lattice–like structures*. In KRIS können aufgrund dieser Ansätze Regeln formuliert werden.

4.1 Halbordnung

Definition: Eine Menge $V = \{a, b, \ldots\}$ mit zwei zweistelligen Operationen $\sqcup$ und $\sqcap$ heißt in Bezug auf diese Operationen ein Verband[102], wenn die folgenden Axiome gelten[103]:

1) Kommutativgesetze

   (a) $a \sqcap b = b \sqcap a$
   (b) $a \sqcup b = b \sqcup a$

2) Assoziativgesetze

   (a) $(a \sqcap b) \sqcap c = a \sqcap (b \sqcap c)$
   (b) $(a \sqcup b) \sqcup c = a \sqcup (b \sqcup c)$

3) Konkatenation

   (a) $a \sqcap (a \sqcup b) = a$
   (b) $a \sqcup (a \sqcap b) = a$

[102] Verband im Englischen *lattice (structure)*. Der Name Verband stammt von Fritz Klein–Barmen.
[103] siehe Hans Hermes, [65], 1967, Seite 1.

Verbände unterliegen dem Dualitätsprinzip. Die duale Aussage eines Satzes der Verbandstheorie ist wieder ein Satz der Verbandstheorie[104].
Gesetz der Idempotenz:

1) $a \sqcap a = a$
2) $a \sqcup a = a$

Jeder Verband ist eine Halbordnung, aber nicht umgekehrt.

Definition: Eine Menge H, in der eine zweistellige Relation $\Subset$ erklärt ist, heißt in Bezug auf diese Relation eine Halbordung[105], wenn die folgenden Axiome gelten, wobei $a, b \in H$[106]:

1) Reflexivität: $a \Subset a$,
2) Transitivität: Wenn $a \Subset b$ und $b \Subset c$, dann $a \Subset c$,
3) Antisymetrie: Wenn $a \Subset b$ und $b \Subset a$, dann $a = b$

Beispiele dazu:

1) Die Menge aller Menschen bilden in Bezug auf die Nachkommenrelation eine Halbordnung. Dabei ist jeder Mensch sein eigener Nachkomme.
2) Die Menge der natürlichen Zahlen ist in Bezug auf die Relation $a \ \ teilt \ \ b$ eine Halbordnung.
3) Ebenso bilden die Elemente von $\mathfrak{P}(m)$ in Bezug auf die mengentheoretische Inklusion $\subseteq$ eine Halbordnung.

[104] Dieser Satz selbst ist kein Satz der Verbandstheorie, vielmehr ist er ein Satz der Meta–Verbandstheorie.
[105] Synonym für *Halbordnung*: *teilweise geordnete (halbgeordnete) Menge*. Im Englischen *partial ordered set*. Ordnungen und Wohlordnungen sind spezielle Halbordnungen. Man findet auch die Begriffe Ordnung und strenge Ordnung.
[106] Äquivalenzrelationen haben die Forderung nach Reflexivität, Symmetrie und Transitivität.

Eine beliebige Teilmenge H' einer Halbordnung H ist in Bezug auf die gleiche Inklusion wieder eine Halbordung. Dieser Übergang von H nach H' heißt Relativierung.
Nützlich sind folgende Begriffe:

- a umfaßt b, falls alle $b \Subset a$.
- b liegt zwischen a und c, wenn $a \Subset b \Subset c$.
- a ist echt in b enthalten, falls $a \Subset b$ und $a \neq b$.
- Wenn a echt in b liegt, und es kein c zwischen a und b gibt, dann heißt a *unterer Nachbar* von b und b *oberer Nachbar* von a. a und b sind *benachbart*.

Auch in den Halbordnungen gilt das Dualitätsprinzip.
Eine Halbordnung heißt Ordnung oder Kette, falls a und b vergleichbar sind, d.h. $a \Subset b$ oder $b \Subset a$ gilt[107]. Ein Beispiel sind die natürlichen Zahlen in Bezug auf die Relation $\leq$.
Ein Element einer Halbordnung, das kein anderes umfaßt, heißt minimal. Im Beispiel 1 sind dies alle kinderlosen Menschen. Ein Element, das in jedem Element von H enthalten ist, heißt *kleinstes Element* oder *Nullelement* (0). Sind a und b Nullelemente so gilt $a = b$.
Die oberen Nachbarn von 0 heißen *Atome*, die oberen Nachbarn der Atome *Hyperatome*.
Dazu dual ist der Begriff des *maximalen, größten Elements* oder 1. Nicht jede Halbordnung beinhaltet 0 und 1.
Eine nichtleere endliche Halbordnung H besitzt wenigstens ein maximales Element. Jede nichtleere endliche Halbordnung läßt sich durch ein Diagramm[108] darstellen. Die aus der realen Analysis bekannten Begriffe der *Schranke* und *Grenze* lassen sich mit denselben Definitionen auch für beliebige Halbordnungen bilden.

[107] Das Gesetz der Trichotomie bedeutet, daß genau eine der folgenden Relationen gilt: $a < b$, $a = b$ oder $a > b$.
[108] Dieses Diagramm ist auch bekannt unter dem Namen Hasse–Diagramm.

4.2 lexikalisches Programm

In Konstanz wurde von der Forschergruppe „Das Lexikon der Organisation der Sprache“ ein lexikalischer Zugang zu Verben der Bewegung[109] versucht. Im Folgenden wird diese Darstellung kurz erläutert. Begriffe zur Darstellung der Welt, die auf Objekten mit Eigenschaften basieren, können in Bäumen oder *lattice–like structures*[110] beschrieben werden. Eine Katze ist leicht als Säugetier zu identifizieren. Säugetiere wiederum gehören zu den Warmblütern. Bei Handlungen ist die Identifikation ungleich schwieriger.
Auch Verben lassen sich in Gruppen zusammenfassen. Das Verb „gehen“ beschreibt eine Bewegung und ist aus der Gruppe des Objekttransfers, das zur Familie der Statusänderungen gehört. Im Unterschied zu Substantiven besitzen Verben eine interne Struktur, die nicht aus ihrer Position in solch einem Baum abgeleitet werden kann. Auch über die Katze kann man mehr wissen, als sich aus ihrer Einordnung in so einem Baum ableiten läßt. Bei Verben ist sofort zusätzliches Wissen nötig, um eine Einordnung zu begründen. Ein anderes Problem ist die extreme Kontextabhängigkeit von Handlungen. Es scheint keine einfache Repräsentation von Verben zu geben, die wie ein logisches Prädikat eine Basis für Mengen von Ableitungen bildet[111]. Wenn ein Verb getestet wird, läßt sich über die Bedeutung des Verbs eine zugehörige Menge von Implikationen finden. Aus dieser Menge läßt sich dann eine als primär ausfiltern. Die anderen Bedeutungen lassen sich durch Erweiterungen oder Verkürzungen ableiten[112]. Auch bei unterschiedlichen Begriffen lassen sich ähnliche Strukturen aufzeigen[113]. Dies läßt vermuten, daß es möglich ist, eine Kerndefinition der Verben zu finden, die nur eine Menge ihrer abstrakt formulierten, logischen Implikationen enthält. Ein Modell der semantischen Struktur muß aus dieser

[109] siehe Bruce Mayo, [82], 1991.
[110] siehe Ronald Brachman, [20], 1985.
[111] Kathleen Dahlgren löst dieses Problem statistisch (siehe Kathleen Dahlgren, [37], 1988, insbesondere Kapitel 4).
[112] siehe Peter Pause, [94], 1991.
[113] siehe dazu Seite 36.
Ein Beispiel ist „Eintreten in einen Raum“ und „Eintreten in eine Partnerschaft“. Beide Fälle lassen ähnliche und widersprüchliche Formulierungen zu.

Menge der Regeln, die Implikationen ableiten können, die alle Kombinationen dieser Argumente erzeugen[114]. Wenn das für einige Argumente oder Interpretationen von Verben scheitert, kann eine Doppelbedeutung angenommen werden[115]. Es ist sicher unmöglich, alle semantischen Eigenschaften eines Verbs darzustellen. Aber die wichtigsten Eigenschaften können auch durch ein Computerprogramm aufgezeigt werden.
Sinnvoll ist eine Unterteilung in:

- grammatisches Lexikon Modul (Regel der Grammatik),
- semantisches Erinnerungsmodul (semantischer Kontext),
- semantisches Lexikonmodul (resultierende Fakten der semantische Interpretation)[116],
- Weltwissenmodul (sprachunabhängiges Wissen) und
- linguistisches Grundmodul (linguistisches Wissen, das unabhängig von Erfahrung und Sprachlernen ist).

In der Hierarchieebene ist das linguistische Grundmodell ganz oben, gefolgt vom Modul des Weltwissens. Die Idee ist, daß bei Änderungen in der höheren Hierarchieebene, die anderen Ebenen auch geändert werden müssen. Eine niedrigere Ebene beeinflußt eine höhere nicht. Diese Strukturierung ist jedoch nicht durchgehend aufrecht zu halten, da in der Linguistik die Grenzen zwischen den Ebenen diffus sind.
Zur Klassifizierung von Verben werden in der Linguistik auch die Abstände zwischen Worten gezählt. Der Abstand zweier Worte entspricht der Anzahl der Worte, die zwischen den betrachteten Worten liegen. Dabei gilt, daß jedes Verb der Name eines Begriffs ist.

[114] siehe Bruce Mayo, [82], 1991, Seite 5.
[115] siehe *two–level semantic* bei Manfred Bierwisch, [18], 1983.
[116] Viele semantische Interpretationen basieren auf Wissen über die Eigenschaften der betrachteten lexikalischen Begriffe. Ein semantischer Zugang verwendet semantische Grundprädikate wie $in(x,y)$, $prior\_to(a,b)$.

Zwei identische Verben beschreiben prima facie zwei verschiedene Objekte (Ereignisse). Erst in der Resolution können sie unter Umständen identifiziert werden. Eintreten wird zum Beispiel repräsentiert durch:

- *A* tritt ein, wenn ...
- *A* betritt *B*, wenn ...

Eine dazugehörende Regel ist:

a tritt ein, wenn

a zunächst nicht in *in_rel* zu *x*, und danach in *in_rel* ist mit *x*.

In Prolog läßt sich dies folgendermaßen dargestellen:

```
eintreten(a):-before(not(in_rel(a,x))),in_rel(a,x).
```

Beim Verlassen eines Raums ist die Reihenfolge genau andersherum:

```
verlassen(a):-before(in_rel(a,x)),not(in_rel(a,x)).
```

Ein Raum muß erst einmal betreten worden sein, um ihn wieder verlassen zu können. Diese Information muß sich als Fakt in der Datenbasis befinden. Die gleiche Relation kann beim „Aufwachen“ verwendet werden[117]. Aber mit „betreten des Raums“ ist eine Raumveränderung ausgedrückt. Eine Lösung ist, jedem Verb einen Baum von semantischen Kategorien[118] mitzugeben.

```
1 verlassen:-
2   assert(kausal(x,verlassen),nil),
3   assert(art1(V1),nil),
4   assert(raumsemantisch(before(in_rel(a,x)),not(in_rel(a,x)),nil)).
```

Der kausale Grund könnte in der Person selbst liegen und `art1` könnte eine Geschwindigkeit sein. Andere Prädikate sind:

[117] Ein Beispiel für diese Relation ist: $before(in\_rel(a,schlaf),not(in\_rel(a,schlaf)))$.

[118] Welche Prädikate in dieser Beschreibung verwendet werden, ist abhängig von der verwendeten Sprache (siehe Götz Wienold, [121], 1989).

- `komplement`,
- `pfad_topology`,
- `relational`

Peter Pause[119] verwendet für Handlungen in Prolog folgende Einteilung der Prädikate:

`ACT(x,P)` : *P* ist eine Handlung des Agenten *x*.

`CHANGE(P)` : Dieses Prädikat beschreibt den Übergang in den Zustand *P* nach Ausführen der Handlung *s*.

`BEGIN(P)` : Dieser Operator beschreibt den Beginn von *P* und ist eine Spezifikation von `Change`. Dabei erfolgt ein Übergang von $\neg P$ nach *P*.

`END(P)` : Damit wird das Ende von *P* bezeichnet. `END` ist wie `BEGIN` ein Phasenquantifizierer, nur in umgekehrter Richtung.

`CAUSE(P)` : `CAUSE(P)` hat *P* als seine Ursache und `CAUSE(P)` ist das Ergebnis im kausalen Konzept.

`ET(P,Q)` : Diese Formel repräsentiert das komplexe Prädikat *P* und *Q*

`SPEC(P,Q)` : Der Operator spezifiziert das Prädikat *P* durch *Q*.

[119] siehe Peter Pause, [94], 1991, Seite 33.

4.3 Propositionen

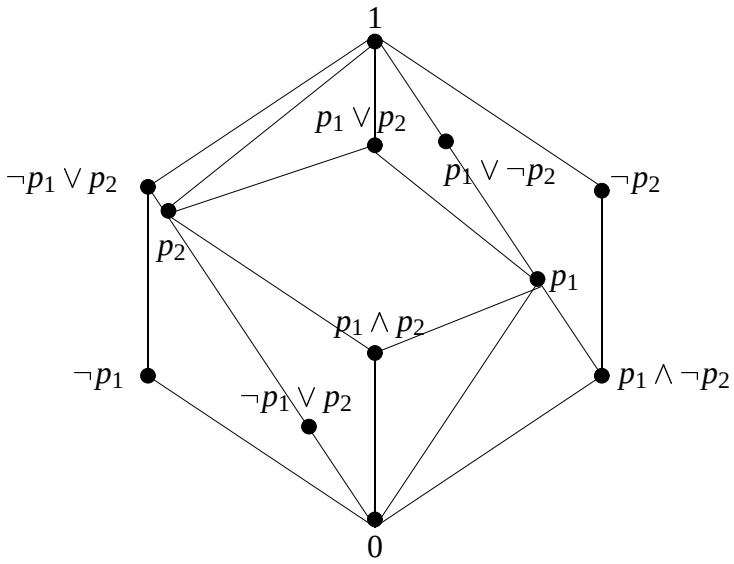

Abbildung 3: Propositionenraum für zwei Handlungen p und p'.

Propositionen bezeichnen abstrakte Entitäten, die Inhalte oder Objekte von mentalen Akten oder Sprechakten sind[120]. Der Begriff des „Satzes“ im Deutschen ist zweideutig. Entweder ist das Sprachgebilde oder der Inhalt eines Urteils von Bedeutung. Die englische Sprache unterscheidet zwischen ‚*sentence*‘ und ‚*proposition*‘. Eine „Proposition“ ist also der Inhalt oder Gehalt eines Satzes[121]. Propositionen sind Klassen von bedeutungsgleichen Sätzen. Ein Ereignis wird durch alle bedeutungsgleichen Sätze, von denen einer das Ereignis beschreibt, dargestellt. Handlungen sind spezielle Ereignisse.

Eine Menge P von Propositionen, die mit einer Implikation $\preceq$ die folgenden sechs Bedingungen erfüllt, heißt Propositionenraum. In einem Propositionenraum können auch Handlungen repräsentiert werden, die keine sprachliche Darstellung haben. Propositionen sind also nicht unbedingt Sätze, sondern auch Objekte aus einem Propositionenraum.

[120] siehe Jürgen Mittelstraß, [87], 1995, Seite 364.
[121] siehe Richard C. Jeffrey, [67], 1965, Seite 65.

Die minimale, formale Struktur eines Propositionenraums besteht aus einer zweistelligen Relation $\preceq$ auf einer Menge P von Propositionen. „$p \preceq p'$“ bedeutet: „p impliziert p'“[122].

> „Wenn die Bedeutung von Satz s, die von Satz s' impliziert und umgekehrt, dann haben s und s' die gleiche Bedeutung. Das heißt, daß die von s und s' erzeugten Propositionen (Satzmengen) p und p' identisch sind.“[123]

Für Propositionen, sind folgende Annahmen plausibel:

(1) wenn $p \preceq p'$ und $p' \preceq p$, dann gilt $p = p'$.

Die Implikation ist reflexiv und transitiv:

(2) $p \preceq p$ (Reflexivität),

(3) wenn $p_1 \preceq p_2$ und $p_2 \preceq p_3$, dann ist auch $p_1 \preceq p_3$ (Transitivität).

Somit bildet die Menge P der Propositionen mit der Implikation $\preceq$ eine Halbordnung.
Die Satzmenge für einen Propositionenraum besteht meist aus logisch komplizierten Sätzen, für die es in der Regel kein Routineverfahren gibt, mit dem entschieden werden kann, ob zwei Sätze äquivalent sind[124]. Dazu einige kurze Bemerkungen zur Verbandstheorie[125], wobei die Existenz von Konjunktion, Disjunktion und Negation vorausgesetzt wird. In der Menge P existieren auch logisch wahre und falsche Propositionen.

- Das Infimum zweier Propositionen p und p' ist die größte untere Schranke $p \sqcap p'$.
- Das Supremum zweier Propositionen p und p' ist die kleinste obere Schranke: $p \sqcup p'$.
- Das Infimum über alle Propositionen wird mit 0, das Supremum mit 1 bezeichnet.

[122] Diese Implikation ist schwächer als die logische Implikation!
[123] Wolfgang Balzer, [6], 1993, Seite 94.
[124] siehe Richard C. Jeffrey, [67], 1965, Seite 79.
[125] Eine genaue Erläuterung erfolgt in Abschnitt 4.1 ab Seite 31.

- Existiert zu einer Proposition p eine Proposition q, sodaß $p \sqcap q = 0$ und $p \sqcup q = 1$, so ist q ein Komplement oder eine Negation von p und wird mit $\neg p$ bezeichnet.

Diese Operationen entsprechen den logischen Junktoren „und“, „oder“ und „nicht“.
„0“ und „1“ entsprechen den logisch falschen bzw. wahren Sätzen.
Deswegen gelten auch folgende Gesetze:

(4) Für je zwei Propositionen $p,\ p'$ in P existieren deren Infimum und Supremum,

(5) 0 und 1 existieren in P,

(6) Zu jeder Proposition p in P gibt es genau eine Proposition $\neg p$ in P,
für die $p \sqcap \neg p = 0$ und $p \sqcup \neg p = 1$ ist.

Verbände in denen (1) bis (6) gelten, heißen komplementär. Desweiteren sollen Supremum und Infimum das Distributivgesetz erfüllen:

für alle $p_1,\ p_2,\ p_3$ in P gilt:

$$p_1 \sqcap (p_2 \sqcup p_3) = (p_1 \sqcap p_2) \sqcup (p_1 \sqcap p_3)$$

$$p_1 \sqcup (p_2 \sqcap p_3) = (p_1 \sqcup p_2) \sqcap (p_1 \sqcup p_3).$$

In einem endlichen Propositionenraum ist auch die Vollständigkeit und Existenz von Atomen garantiert. Die Implikation „$\preceq$“ bietet bei Handlungen zwei Deutungsmöglichkeiten. Einerseits kann eine übergeordnete Handlung eine andere enthalten (Beispiel: „Das Darlehen von der Bank zu nehmen“ ist ein Bestandteil der Handlung „ein Haus bauen“). Andererseits kann auf der Bedeutungsebene eine Handlung die andere beinhalten (Beispiel: „Hans zu küssen“ impliziert „Hans berühren“). Das Infimum zweier Handlungen ist die Handlung, bei der beide ausgeführt werden. Dabei wird vorausgesetzt, daß die Teilhandlungen „verträglich“ sind.

Die Negation einer Handlung ist das Nichtausführen einer Handlung. Dies kann durchaus aktiv geschehen, wenn z.B. das Nichtstun von England und Frankreich bei Hitlers Besetzung des Rheinlandes[126] betrachtet wird. Das Supremum ist die kleinste Handlung, die von beiden impliziert wird.

Bei dem intendierten System werden endlich viele Propositionen verwendet. Diese Liste wird nun durch Suprema, Infima und Komplemente angereichert, bis keine neuen Propositionen entstehen.

Hat man zwei Handlungen p und p', zeigt die Abbildung 3 auf Seite 38 den Propositionenraum. Bei endlich vielen Ausgangspropositionen steigt die Anzahl möglicher Kombinationen exponentiell. Wir haben nun eine Menge H von Handlungen und den sie repräsentierenden Propositionenraum $\langle P, \preceq \rangle$.

Propositionenräume sind vielfältig. Die Propositionenräume von Personen unterscheiden sich. In der Postmoderne unterscheiden sich die Propositionenräume einer Person je nach Rollenverhalten.

Bei Handlungen wird unterschieden zwischen Typ und Vorkommnis.

> „Ein Vorkommnis ist eine Handlung in ihrer konkreten Einmaligkeit, während ein Handlungstyp als eine Klasse von ähnlichen Vorkommnissen aufgefaßt werden kann (sie steht für das, was allen ihren Vorkommnissen gemeinsam ist). Eine solche Klasse wird durch eine konkrete Handlung a erzeugt, nämlich als Klasse aller zu a ähnlichen Handlungen.“[127]

Handlungstypen sind nicht notwendig disjunkt. Zustände können als Grenzfälle von Handlungen betrachtet werden:

> „Zur Bildung von Präferenzen zwischen Handlungen ist es oft nötig, die Ergebnisse der Handlung miteinzubeziehen. Indem wir solche Ergebnisse, die in den meisten Fällen Zustände sind, als Grenzfälle von Handlungen auffassen, erzielen wir eine homogene und einfache Darstellung.“[128]

[126] Dieses Beispiel wurde entnommen aus: Wolfgang Balzer, [6], 1993, Seite 97.
[127] Wolfgang Balzer, [6], 1993, Seite 101.
[128] Wolfgang Balzer, [6], 1993, Seite 103.

Für Handlung *a*, *b* und einen Akteur *i* wird definiert:

(7) *a verursacht* nach *i*'s Überzeugung *b teilweise* per Definition
genau dann, wenn es Propositionen p_a, p_b gibt, sodaß gilt:
in *i*'s internem Modell wird *a* durch p_b repräsentiert und
p_a verursacht nach *i*'s Überzeugung p_b teilweise.

Dies heißt nichts anderes als „*i* glaubt, daß Handlung *a* eine Teilursache von Handlung *b* ist".

4.4 Nichtausführung von Handlungen

Ein gerichteter Graph ist eine Menge mit einer endlichen zweistelligen Relation.

> „Ein Graph ist definiert als eine Menge irgendwelcher Objekte (Punkte, Knotenpunkte oder Ecken des Graphen genannt) und einer zweistelligen Relation, die zwischen den Elementen (Punkten) der Menge erklärt ist. Stehen zwei Punkte (Objekte) in der Relation, so sagt man, sie liegen auf derselben Linie oder Kante des Graphen.“[129]

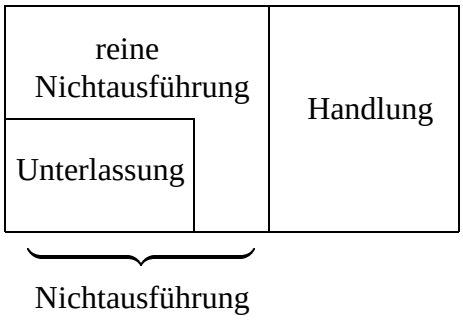

Abbildung 4: Nichtausführung von Handlungen,
nach Hans Lenk, [78], Seite 152.

Manche Autoren behandeln auch Graphen mit Elementarschleifen, d.h. die Relation muß nicht irreflexiv sein. Wird die Beschränkung der Irreflexivität aufgehoben, können Ordnungen, Quasiordnungen, Ketten, Halbverbände, Verbände und andere Strukturen der Handlungslogik graphentheoretisch untersucht werden. Die „Dadurch–daß–Relation“ von Goldman[130] bezeichnet die Stufengeneralisierung zwischen Handlungsvorkommnissen (*act–tokens*) oder Handlungstypen (*act–types*). Handlungsvorkommnisse beinhalten im Gegensatz zu Handlungstypen Angaben über Akteur, Zeit und Ort[131].
Handlungseigenschaften (*act–properties*) werden durch generelle Handlungstypen beschrieben[132]. Die Stufengenerierung stellt Goldman in Handlungsdiagrammen dar, die sich topologisch aus Elementarhandlungsvorkommnissen aufbauen. Lenk beschränkt sich auf Handlungsvorkommnisse, bemerkt aber, daß sich dieses Konzept auch auf Handlungstypen und Handlungseigenschaften erweitern läßt. Goldman spricht von Handlungsbäumen (*act–trees*). Graphentheoretisch handelt es sich dabei nicht um Bäume, da Zyklen und zusammengesetzten Generierungen auftreten können. Die „Dadurch–daß–Relation“ beschreibt einen Handlungskomplex strukturell. Nimmt man Goldmans Strukturkom-

[129] Hans Lenk, [78], 1980, Seite 138.
[130] siehe Abschnitt 3.2 auf Seite 29.
[131] siehe konkrete Handlung in Abschnitt 4.3, Seite 41.
[132] Bei Goldman werden Handlungseigenschaften und Handlungstypen gleichgesetzt. Dazu siehe Alvin Goldman, [58], Seite 10.

plex in Bezug auf die einzelnen Handlungsvorkommnisse, ergibt sich ein endlicher und asymmetrischer Digraph mit mindestens einem Sendepunkt, dem Elementarhandlungsvorkommnis. In der Graphendarstellung läßt sich zeigen, daß irreflexive transitive Relationen auch asymmetrisch sind. Transitive Graphen dürfen keine Zyklen enthalten. In Handlungsbäumen folgt aus der Forderung nach einem zyklenfreien Handlungssystem, daß keine zusammengesetzte Handlungen auftreten dürfen. Kann der irreflexive transitive Digraph auch konnex (zusammenhängend) konstruiert werden, spricht man von Reihen oder strikter Ordnung. Haben Handlungsvorkommnisse identische Vorgänger und Nachfolger, erhält man unter Umständen eine Verbandsstruktur. Zu je zwei Handlungsvorkommnissen gibt es einen gemeinsamen oberen Vereinigungs– und einen gemeinsamen unteren Verzweigungspunkt. Je zwei Punkte haben also ein Supremum oder Infimum.
Die Verbandsstruktur scheint jedoch die Ausnahme zu sein. Meist handelt es sich um einen Halbverband. Zu zwei Punkten findet sich in Richtung des Erzeugungspunktes ein Verzweigungspunkt.
Zeitlich aufeinanderfolgende Handlungen werden hierbei noch nicht berücksichtigt. Unabhängig voneinander stattfindende gleichzeitige Handlungen können nicht auf demselben Graphen angeordnet werden. Werden zwei Handlungsvorkommnisse auf demselben Handlungsgraphen angeordnet, so sind sie gewissermaßen partiell „identisch“. Handlungsvorkommnisse desselben Handlungsgraphen sind Elemente derselben Komplexhandlung. Sie ist eine abstrakte Äquivalenzrelation, die zur Klassenbildung von Handlungsaussagen genutzt werden kann. Verbände haben eine strukturelle Geschlossenheit, die zur Erfassung spezifischer Zusammenhänge von Handlungsaussagen besonders geeignet ist[133]. Diese Eignung kommt bei der Darstellung der Negationen von Handlungsaussagen zum Tragen.
Die wahrheitsfähigen (mit einem Wahrheitswert belegbaren) Aussagen über Handlungen bilden bezüglich des „und“ und des „oder“ einen Verband, wenn die Handlungsaussagen,

[133] siehe Hans Lenk, [78], 1980, Seite 146.

die mit diesen Verknüpfungszeichen gebildet werden, wieder Handlungsaussagen sind. Der Verband ist ein boolescher Verband, wenn die Negation einer Handlungsaussage als Komplement, sowie „das Wahre“ und „das Falsche“ als universelles, beziehungsweise Nullelement eingeführt werden[134].

Die Verknüpfung zweier Handlungen kann einerseits auf der Satzebene mit dem logischen „$\wedge$“ ausgeführt werden oder es kann eine neue Handlung postuliert werden, die beide Handlungen enthält. Zur Verbindung zweier Handlungsterme (Pläne, Leistungen, Forderungen) zu neuen Handlungstermen[135] werden die Operatoren „&“ für „und“, sowie „∇“ für „oder“ definiert. Das neue Zeichen für die Negation (logisch: „$\neg$“) ist „$-$“. Die Quasiordnung der Erfüllungsbeziehung $A_1 \boxed{\leq} A_2$ erhält folgende Interpretation:

Die Ausführung der bezeichneten Leistung (Handlung) A_1 erfüllt die Forderung (Plan) A_2 und bringt diese dadurch zum „Verschwinden“[136]. Wechselseitige Erfüllung entspricht der Gleichwertigkeit: $A_1 \boxed{\equiv} A_2$. Diese Erfüllungsbeziehung ist reflexiv und transitiv. Die kommutative und assoziative Verknüpfung $A_i \boxed{\leq} (A_i \nabla A_k)$ bedeutet, daß mindestens eine der Teilleistungen die komplexe Alternativforderung erfüllt. Ebenso erfüllt jede konjunktive Gesamtmaßnahme die zugehörige Teilforderung in $A_i \& A_k \boxed{\leq} A_i$. Nun ist $A_i \nabla A_k$ das Supremum und $A_i \& A_k$ das Infimum von A_i und A_k bezüglich $\boxed{\leq}$ (ohne Berücksichtigung der Gleichwertigkeit). Um einen distributiven Verband zu erhalten, wird auch die Distributivitätsregel angenommen: $A_i \& (A_k \nabla A_l) \boxed{\leq} (A_i \& A_k) \nabla (A_i \& A_l)$. Mit der trivialen Forderung $\top$, die durch jede Leistung erfüllt wird und der absurden Leistung $\bot$, die jede Forderung erfüllt, erhält man einen distributiven Verband mit Einselement $\top$ und Nullelement $\bot$. Jede Handlungskonjunktion mit dem absurden Plan ergibt den absurden Plan und jede Handlungsalternativbildung mit der trivialen Forderung ergibt wiederum die triviale Forderung. Handlungskonjunktion (Handlungsalternativbildung) A_i mit $\top$ ($\bot$) ergibt A_i. Gilt $A_1 \& A_2 \boxed{\equiv} \bot$ und gleichzeitig $A_1 \nabla A_2 \boxed{\equiv} \top$, so ist A_1 ein Komplement von A_2. $\top$ und $\bot$

[134] siehe Georg Henrik von Wright, [124], 1980, Seite 24.
[135] siehe Hartmann J. Genrich, [52], 1980, Seite 107.
[136] siehe Hans Lenk, [78], 1980, Seite 148.

sind Komplemente voneinander. In distributiven Verbänden hat ein Element A_1 höchstens ein Komplement[137], das Supremum aller zu diesem Element inkompatiblen Leistungen. Das Komplement von A_i wird mit $-A_i$ gekennzeichnet.
Die Komplementhandlung entspricht lediglich dem allgemeinen Nichtausführen (*neglection*) einer Handlung. Eine spezifische intentionale Unterlassung (*forbearance, preventive action*) ist dabei nicht beinhaltet. Die Nichtausführung einer Nichtausführung ergibt die Handlung (*tertium non datur*).
Die gewollte Unterlassung beinhaltet die bloße Nichtausführung aber nicht umgekehrt[138]. Somit gibt es hier ein drittes: die Nichtausführung ohne Unterlassung.
Wie in Abbildung 4 auf Seite 43 graphisch dargestellt, gibt es zweierlei boolesche Verbände:

- Handlungen und ihre Nichtausführung, mit der schwachen Negation und
- Unterlassung innerhalb der Teilmenge von Handlungen samt Unterlassung.

Nach Lenk[139] ist eine Lösung dieses Problems das Arbeiten mit einer schwachen ($-$) und einer starken ($\sim$) Negation. Die starke Negation entspricht der bewußten Unterlassung.

[137] siehe Hans Hermes, [65], 1967, Seite 50.
[138] Die absichtliche Unterlassung einer Handlung setzt voraus, daß diese ausgeführt werden könnte. Nach von Wright kann ein Analphabet das Lesen nicht unterlassen. Dazu siehe Georg Henrik von Wright, [124], 1980, Seite 26.
[139] siehe Hans Lenk, [78], 1980, Seite 152.

5 Die Sprache des Computers

Urs Egli[140]unterscheidet zwischen

- imperativen Programmiersprachen:
  Befehle müssen Schritt für Schritt eingegeben werden. Diese Befehle werden sequentiell abgearbeitet. Beispiele für diese Sprachen sind Basic, Fortran und Pascal.
- applikativen Programmiersprachen:
  Hier wird von Funktionen ausgegangen, die der Computer verarbeiten muß. Ein Beispiel ist Lisp.
- deklarativen Programmiersprachen:
  Es werden Fakten und Regeln eingegeben. Ein Beispiel ist Prolog.

5.1 Lisp

Der Name steht für „List Processor“ . Diese Sprache, die in den 50er Jahren von John McCarthy[141] in Stanford entwickelt wurde, findet immer noch Verwendung bei der Programmierung von AutoCAD und des Editors Emacs. Der Vorteil von Lisp ist die Einfachheit. Es gibt nur das Konstrukt der Liste. Manipulationen mit solchen Listen sind im Lambda–Kalkül von Alonzo Church[142] beschreibbar. Unterarten des Programms sind „Common Lisp“ und „Scheme“. Für Scheme existiert eine vollständige formale semantische Definition. Um Lisp in Emacs aufzurufen erfolgt die Eingabe „M-x lisp-interaction-mode“. Die dort gemachten Eingaben gelangen mit „Ctrl-j“ zur Ausführung[143]. Beispiele für soziale Simulationen in Lisp finden sich im *Journal of Artificial Societies and Social Simulation*[144].

[140]siehe Urs Egli, [41], 1992, Seite 14.
[141]siehe John McCarthy, [83], 1996.
[142]siehe Alonzo Church, [31], 1956.
[143]siehe Matthias Kalle Dalheimer, [38], 1997, Seite 161f.
[144]siehe Nigel Gilbert, [56], 1999.

5.2 SWI–Prolog

Das für KRIS verwendete SWI-Prolog ist eine Prologimplementierung von Jan Wielemaker der Universität Amsterdam. Eine bemerkenswerte Eigenschaft ist die Vorcompilierung des Codes zur Beschleunigung der Programmausführung. SWI-Prolog ist auch für MS-Windows und OS/2 erhältlich. Dieses Programm ist somit weitgehend plattformunabhängig[145]. Das verwendete SWI-Prolog verwendet die logische Datenbasissicht. Veränderungen an einer Prozedur werden während der Verarbeitung nicht bemerkt[146]. Verschiedene Prologdialekte können jedoch gleichzeitig mehrere Prozesse verarbeiten. Am geeignetesten erscheinen Multiprozessorsysteme[147]. Aber auch sequentielle Prologsprachen, die mehrere Prozesse gleichzeitig verarbeiten können, sind vorhanden[148].

5.3 XPCE

XPCE ist ein Programm, mit dem grafische Benutzeroberflächen entwickelt werden können. Es existieren Bindungen für C++, Lisp und eben Prolog. Die Programmierung erfolgt im Gegensatz zu Prolog objektorientiert. Es existieren vier Prädikate:

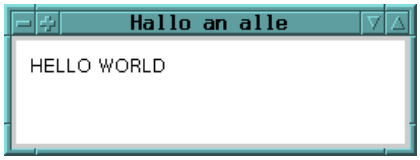

Abbildung 5: „Hallo Welt“.

- `new/2`,
- `send/[2-12]`,
- `get/[3-13]` und
- `free`.

Ein einfaches Programm wäre folgendes:

[145] siehe Matthias Kalle Dalheimer, [38], 1997, Seite 173.
[146] siehe Ulrich Geske, [55], 1993, Seite 51.
[147] siehe DMASS in Abschnitt 6.4 aus Seite 83.
[148] siehe dazu T-Prolog, CS-Prolog, TC-Prolog usw. bei Ivan Futo, [49], 1990.

```
2       start :-
3         new(P,picture('Hallo an alle')),
4         send(P,display,text('HALLO WELT'),point(10,10)
5         ),
6         send(P,open).
```

Dabei wird das Bildobjekt „Hallo an alle“ angelegt und an die Variable ‚P‘ gebunden. An das Ausgabefenster wird die Nachricht ‚display‘ gesendet. Nun muß das Ganze nur noch auf den Bildschirm gebracht werden.

5.4 Prolog

„Prolog“ steht für Programmierung in Logik. In der Prädikatenlogik werden Objekte als Terme dargestellt.

Diese Darstellung in der Prädikatenlogik entspricht in etwa der Darstellung in Prolog[149]. In Prolog wird die Prädikatenlogik auf eine Klauselform beschränkt. Prolog erfordert Horn–Klauseln, d.h. Klauseln mit höchstens einem positiven Literal[150]. Die Idee zu Prolog stammt von Robert A. Kowalski[151]. Die Umwandlung von Ausdrücken der Prädikatenlogik der ersten Stufe in ein Prologprogramm kann in zehn Schritten[152] vorgenommen werden[153]. Matching entspricht in etwa der Unifikation der Prädikatenlogik.

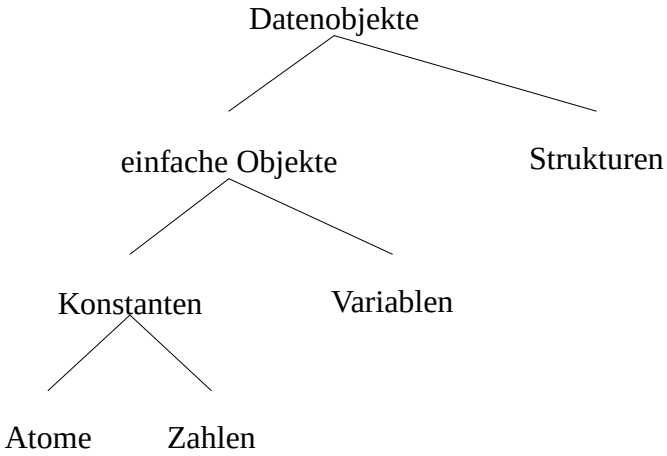

Abbildung 6: Datenobjekte in Prolog.

Ein Beispiel in dem Matchen gelingt und die Unifikation scheitert, ist folgende zirkuläre Struktur:

$$?- \mathtt{X} = \mathtt{f}(\mathtt{X}).$$

Prolog verfolgt einen problemorientierten Ansatz. Im Gegensatz zu anderen Programmiersprachen ist es nicht nötig, einen Algorithmus zur Problemlösung zu finden. Das

[149] siehe Abschnitt 5.4, Seite 50.
[150] siehe Ivan Bratko, [21], 1994, Seite 63f.
[151] siehe Robert A. Kowalski, [73], 1979.
[152] siehe Ulrich Geske, [55], 1993, Seite 15f.
[153] Nach Ivan Bratko, [21], 1994, Seite 64 ist diese Umformung keine Äquivalenzumformung, die auch nicht in Horn–Klauseln, sondern in allgemeine Klauseln umformt.

Problem wird mit Fakten und Regeln beschrieben und Prolog versucht die Lösung zu ermitteln[154].

Traditionelle Programmiersprachen bestehen aus Anweisungen, Prozeduren und logi-

Deutscher Sprachgebrauch		Englischer Sprachgebrauch	
Datenbasis	Fakten	Facts	Clauses
(Wissenrepräsentation)	Regeln	Rules	
Wissensrückgewinnung	Fragen	Questions	Queries

Tabelle 7: Bestandteile eines Prologprogramms nach Armin Ertl, [44], 1988, Seite 21.

schen Datensätzen. Für all diese Strukturen existiert in Prolog nur ein Konstrukt – die Klauseln. Sie können in Prolog je nach Kontext interpretiert werden. Prolog ist so reichhaltig, daß es als intelligente relationale Datenbank[155], als prozeduraler Regelinterpreter (*dynamische Semantik*) oder als Mustererkenner (*pattern matching*) beliebig komplexer Datenstrukturen verstanden werden kann[156]. Jedes Prolog–Programm kann auch als relationale Datenbank aufgefaßt werden, wenn Anfragen an das Programm statt durch Resolution durch Operationen auf eine relationale Datenbank beantwortet werden[157]. Die Antwortmenge von Programmklauseln wird in Form von Relationen abgespeichert. Ein Prologprogramm besteht aus Prädikaten, die wiederum aus Fakten und Regeln bestehen können. Je nach Unifizierung der Argumente beim Aufruf können die Prädikate testen, zugreifen oder generieren. Alles, was nicht in der Faktenbasis vorhanden ist oder aus Regeln abgeleitet werden kann, gilt als falsch.

In Prolog gibt es keine Typ–Deklaration. Prolog kann Variable mit jedem Typ instanziie-

[154] siehe Matthias Kalle Dalheimer, [38], 1997, Seite 169f, sowie Tabelle 8, Seite 53.

[155] Der Programmcode einer komplexen Datenbank in Turbo Prolog findet sich in Tapan Bagchi, [3], 1989, Seite 107 –186.

[156] siehe Peter Schnupp, [112], 1986, Seite 2.

[157] siehe Josef Weingärtner, [119], 1989.

ren und ist deshalb eine typenfreie Sprache[158].

5.4.1 Konjunktionen

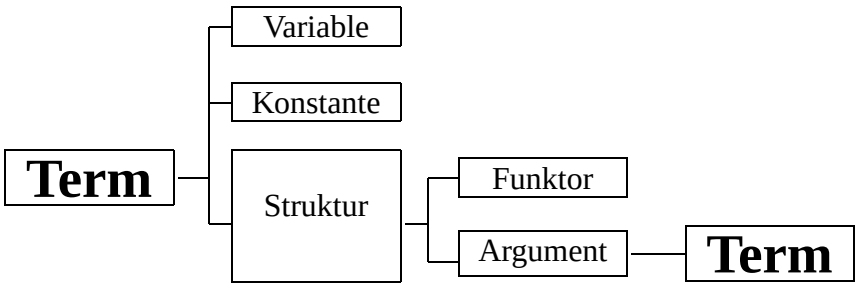

Abbildung 7: rekursive Struktur von Prolog–Termen

nach Peter Schnupp, [113], 1989, Seite 55.

Ein Prologprogramm wird aus Termen gebildet. Terme bestehen aus Konstanten, Variablen oder Strukturen. Im allgemeinen werden Konstanten mit kleinen Buchstaben gebildet. Variablen haben einen Großbuchstaben oder einen Unterstrich zu Beginn der Zeichenfolge, die diese Variable repräsentieren. Den Variablen wird per Unifizierung ein Wert zugeschrieben, das heißt sie werden instanziiert.
Ein zusammengesetzter Term besteht aus einem Funktor, dem in Klammern und durch Kommata getrennt, eine Liste von weiteren Termen folgt. Die Anzahl der Terme heißt Stelligkeit (*arity*) und wird in folgender Form charakterisiert: `relation/x` (Beispiel: `length/2`).
Die Unifizierung in Prolog ist vollsymmetrisch[159]. In Prolog kann nur uninstanziierten

[158] Prolog typt nicht Variablen, sondern Prozeduren. Diese Prozeduren müssen verschiedene Datentypen auch unterschiedlich behandeln.
[159] Bei traditionellen Sprachen wird meist der linken Seite der Wert der rechten Seite zugeschrieben.

Variablen ein Wert zugeschrieben werden.

Der Unterstrich ohne folgende Zeichen bedeutet eine anonyme Variable. Prolog gibt keinen Wert für diese Variable aus[160]. Der Gültigkeitsbereich (Skopus) einer Variablen ist die einzelne Klausel. Eine Struktur besteht aus Komponenten, die zusammengefaßt werden. Die Syntax von Strukturen ist mit der von Fakten vergleichbar. Für die Verbindung von Komponenten stehen die Konjunktion und Disjunktion zu Verfügung. In Prolog schreibt man dafür die Zeichen „ , “ für „und“ sowie „ ; “ für das inklusive „oder“. Ein Prologprogramm besteht aus Fakten, Regeln und Fragen. Fakten sind Daten. Ein Beispiel für ein Fakt ist eine Relation, die zwischen zwei oder mehr Objekten besteht. „*K*uba bedroht Amerika“ ist ein Beispiel für solch einen Zusammenhang. In Prolog wird dies wie folgt geschrieben: `bedroht(Kuba,Amerika).` . Strukturen erhält man aus einfachen Fakten, die näher spezifiziert werden. Beispiel: „*K*uba bedroht große Länder“ in Prolog: `bedroht(Kuba,laender(gross)).` . In der Datenbasis vorhandene Fakten und Regeln können so verstanden werden, daß sie durch die logische Verknüpfung „oder“ verbunden sind. Regeln sind „wenn dann“ Beziehungen, die in Prolog mit dem Zeichen :− ausgedrückt werden. Ein Beispiel ist „*K*uba bedroht X, wenn X Kuba bedroht.“. In Prolog wird dies zum Beispiel folgendermaßen ausgedrückt: `bedroht(kuba,X) :- bedroht(X,kuba).`. Fakten sind identisch mit Regeln, die keinen

(1)	Datenbasis (Fakten und Regeln) einlesen
(2)	Frage einlesen
(3)	Frage bearbeiten
(4)	Antwort ausgeben
(5)	Wiederholen ab Schritt (2)

Tabelle 8: Abarbeitung eines Prologprogramms
nach Armin Ertl, [44], 1988, Seite 20.

[160] siehe Ivan Bratko, [21], 1994, Seite 32.

rechten Teil besitzen[161]. Alle Regeln und Fakten, deren linker Teil übereinstimmt, bilden ein Prädikat[162]. Fragen beginnen mit einem Fragezeichen: $?-\texttt{bedroht}(\texttt{kuba},\texttt{X})$. Prolog durchsucht die Faktenbasis nach Informationen, um eine Tatsache zu bestätigen. Die Antwort „nein“ weist nicht automatisch auf eine kontrafaktische Situation hin. Es kann auch bedeuten, daß keine Informationen in der Faktenbasis gefunden wurden[163]. Ist das Wissen in der Datenbasis nicht korrekt, liefert Prolog falsche Antworten. Die Beschränkung auf in der Datenbasis vorhandenes Wissen wird als *closed world assumption* bezeichnet[164]. Eine Abarbeitung von Regeln (*Resolution*) liefert das Ziel (*goal*). Dazu errechnet Prolog die Werte (*Unifizierung*).

5.4.2 Listen

Eine spezielle Struktur in Prolog ist die Liste. Die Liste ist eine geordnete Menge von Elementen. Sie ist entweder leer oder besteht aus Kopf und Fuß. Die leere Liste wird mit eckigen Klammern geschrieben: [ ].
Eine Liste hat die rekursive Form: $\texttt{L} = [\texttt{Kopf} \mid \texttt{Schwanz}]$.
Der Längsstrich heißt Restlistenoperator. Eine Ausnahme ist die leere Liste. Die leere Liste ist nicht mit diesem Schema unifizierbar und gehört auch nicht zu den Strukturen, sondern ist ein Atom. Es können mehr als ein Element durch den Restlistenoperator abgetrennt werden ($\texttt{L} = [\texttt{A},\texttt{B},\texttt{C} \mid \texttt{Schwanz}]$).
Die Liste $[\texttt{a},\texttt{b},\texttt{c}]$ kann also alternativ
$[\texttt{a} \mid [\texttt{b},\texttt{c}]] = [\texttt{a},\texttt{b} \mid [\texttt{c}]] = [\texttt{a},\texttt{b},\texttt{c} \mid [\ ]\ ]$ geschrieben werden[165].

[161] Die Abfrage $\texttt{clause}(\texttt{R},\texttt{true})$. ist bei jedem Fakt R erfolgreich.
[162] siehe Armin Ertl, [44], 1988, Seite 149.
[163] siehe William F. Clocksin, [33], 1994, Seite 6.
[164] siehe Peter Schnupp, [113], 1989, Seite 20.
[165] siehe Ivan Bratko, [21], 1994, Seite 70.

5.4.3 Backtracking

Prolog führt zur Erfüllung von Zielen automatisch Backtracking durch. Dadurch ist der Programmierer davon befreit, explizit Backtracking einzuleiten. Führt eine Bearbeitung zu dem Ergebnis `fail`, setzt der Prologinterpreter auf dem letzten `goal` auf. Dies ist der letzte Wahlpunkt (*choice point*). Alle Variablen nach diesem Wahlpunkt werden zurückgesetzt. Die Variablen sind also wieder uninstanziiert und können erneut unifizieren.

5.4.4 Cut

Der Cut[166] (Schreibweise: !) ist ein Mittel zur Verhinderung von Backtracking. Dadurch kann ein Programm effizienter werden. Der Cut erhöht durch seine schwere Lesbarkeit die Wahrscheinlichkeit von Programmierfehlern. Je nach Stellung im Programm kann der Cut nur prozedurale[167] Bedeutung haben. Ist die Bedeutung des Cut auch deklarativ, wird es schwierig, den Programmablauf nachzuvollziehen[168].

> „Let us call the ‚parent goal' the goal that matched the head of the clause containing the cut. When the cut is encountered as a goal it succeeds immediately, but it commits the system to all choices made between the time the ‚parent goal' was invoked and the time the cut was encountered. All the remaining alternatives between the parent goal and the cut are discarded."[169]

Ein Beispiel für die Verwendung des grünen Cut ist die Maximum–Relation.

```
1 max(X,Y,X) :- X >= Y,!.
2 max(X,Y,Y) :- X < Y.
```

Für die Lösung ist das Weglassen des Cut in der ersten Klausel unerheblich. Mit Cut muß die zweite Lösung nicht mehr berücksichtigt werden, wenn die erste Klausel gelingt. Oft läßt sich ein roter Cut in einen grünen Cut umwandeln, wenn man das Prädikat `not/1`

[166] In der Literatur finden sich sowohl „der Cut" als auch „das Cut". Die meisten Autoren verwenden die erste Möglichkeit.

[167] zu deklarativer und prozeduraler siehe Abschnitt 5.4.7 ab Seite 63.

[168] siehe M. van Emden, [42], 1982, unterscheidet zwischen „grünem" und „rotem" Cut.

[169] Ivan Bratko, [21], 1994, Seite 129.

verwendet. Das Prädikat `not/1` ist eine Verneinung durch Fehler (*negation as failure*) definiert. Bei Erfolg werden alle Variableninstanzierungen (im Gegensatz zu allen üblichen Prolog–Prädikaten) gelöst. Dieses Verhalten von `not/1` kann genutzt werden, um alle vorgenommenen Variableninstanziierungen zu lösen.

```
verarbeite(Term) :-
  clause(Term,_) ,
  not not Term,
  %Variablen sind uninstanziiert!
  weiterverarbeitung_von(Term).
```

Das `not/1` kann also nicht verwendet werden, um herauszufinden, „warum etwas nicht der Fall ist“. Deshalb sollte das Prädikat `not/1` im Regelkörper so spät wie möglich verwendet werden. Korrekt ist die Verwendung von `not/1` in variablenfreien Termen.
Manchmal kann auf die Verwendung des Cut nur schwer verzichtet werden. Ein Beispiel ist das Hinzufügen von Elementen, falls sie noch nicht in der Liste vorhanden sind:

```
add(X,L,L) :-member(X,L),!.
add(X,L,[X | L]).
member(X,[X | L]).
member(X,[Y | L]) :-member(X,L).
```

Ebenso wird bei Ausschluß von Alternativen verfahren[170].

```
g(X,Y) :-f(X,a),!.
g(X,Z) :-z(X,b),!.
g(X,R) :-r(X,c).
```

Für die Klarheit eines Programms ist es sinnvoll, den Cut zumindest optisch zu umgehen. Um einen Prozeduraufruf auf eine einmalige Ausführung zu beschränken, definiert man `once/1`:

[170] Klassifikation in Kategorien, siehe Ivan Bratko, [21], 1994, Seite 133.

```
once(Goal) :-
  Goal,!.
```

Sollen Berechnungen an einer Stelle ergebnislos abgebrochen werden, ist eine Kombination von Cut ! und `fail` nicht zu vermeiden. Ein Beispiel ist das Prüfen, ob alle Elemente verschieden sind:

```
different(X,X) :-!,fail.
different(X,Y).
```

Beim Verfahren *generate and test*[171] ist ein Cut nötig. Der Generator kann beliebig viele Lösungen erzeugen. Ist diese Lösung erfolgreich, ist es unsinnig, weitere Lösungen zu erzeugen.

```
erzeuge_loesung(Loesung) :-
  generiere(Moegl_loesung),
  teste(Moegl_loesung),
  !,
  Loesung = Moegl_loesung.
```

Eine weitere wichtige Anwendung des Cut ist die Ausnahmebehandlung[172].

```
klausel(sonderfall1) :-
  !,
  fail.

klausel(sonderfall2) :-
  write('Dies ist ein Sonderfall'),
  !,
```

[171] siehe Armin Ertl, [44], 1988, Seite 341.
[172] siehe Armin Ertl, [44], 1988, Seite 343.

```
    fail.

klausel(X):-
    !,
    normalfall(X,_,_).
```

Dabei ist zu beachten, daß die „Abfangklauseln“ vor dem Prädikat eingesetzt werden, da sie sonst nicht mehr zum Einsatz kommen.
Wenn ein untergeordnetes Prädikat für die Verifizierung unbedingt nötig ist, sollte links von diesem *k.o.-Prädikat* ein Cut stehen, damit der Interpreter an weiterem sinnlosen Suchen gehindert wird.

```
vater_praedikat:-
    unter_praedikat1,
    !,
    ko_praedikat.
```

Liefert das *k.o.-Prädikat* ein `fail`, versucht der Interpreter weder das Unterprädikat noch das Vaterprädikat zu beweisen[173].

[173] siehe Peter Schnupp, [113], 1989, Seite 29.

5.4.5 Ein– und Ausgabedateien

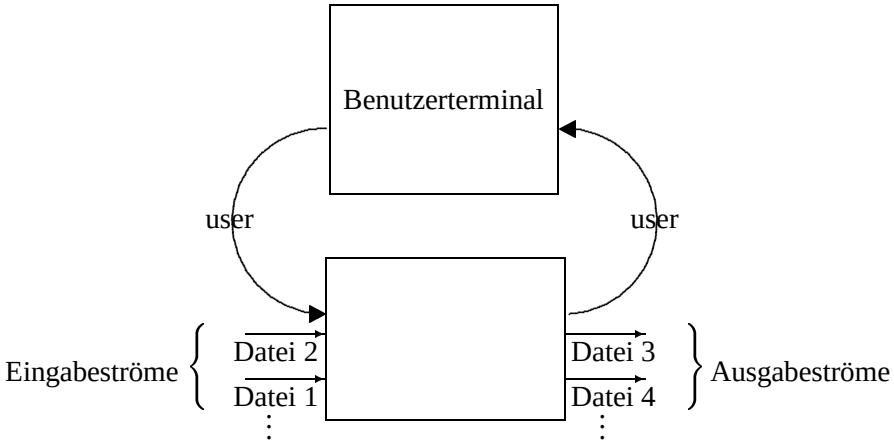

Abbildung 8: Kommunikation zwischen Prolog und Dateien.

Prolog verwendet Ein– und Ausgabedateien. Wird als Datei ‚user‘ angeben, erfolgt die Ausgabe aufs Terminal. Diese Dateien werden auch Ein– und Ausgabestrom genannt. In Linuxterminologie sind dies die Zeichenströme „stdin“ und „stdout“. Die zugehörigen Befehle sind:

- `see`$(<$ `Dateiname` $>)$,
- `tell`$(<$ `Dateiname` $>)$.

Am besten verwendet man folgende Eingabestruktur:

```
seeing(Alte_Eingabe_Datei),
see(Neue_Eingabe_Datei),
...
⋮
seen,
see(Alte_Eingabe_Datei).
```

Soll in der neuen Eingabedatei an der Stelle weitergelesen werden, die erreicht wurde, ist das `seen/0` in der vorletzten Zeile wegzulassen. Analog funktioniert die Ausgabe auf

Dateien, die mit `told/0` abgeschlossen werden müssen. Da `tell(< Dateiname >)` eine vorhandene Datei löscht, ist zum Anhängen von Daten an eine bereits existierende Datei `append(< Dateiname >)` zu verwenden. Weitere Befehle zur Ein– und Ausgabe von Dateien sind:

- `read/1` und
- `write/1`.

Die Befehle

- `assert/1`,
- `asserta/1`,
- `assertz/1` und
- `retract/1`

dienen zur Manipulation der Datenbasis. Damit können Klauseln in die Datenbasis geschrieben oder daraus entfernt werden.

5.4.6 Expertensysteme

Grundsätze der Programmierung sind:

- Korrektheit,
- Effizienz,
- Transparenz,
- Lesbarkeit,
- Modifizierbarkeit,
- Robustheit,
- Dokumentation.

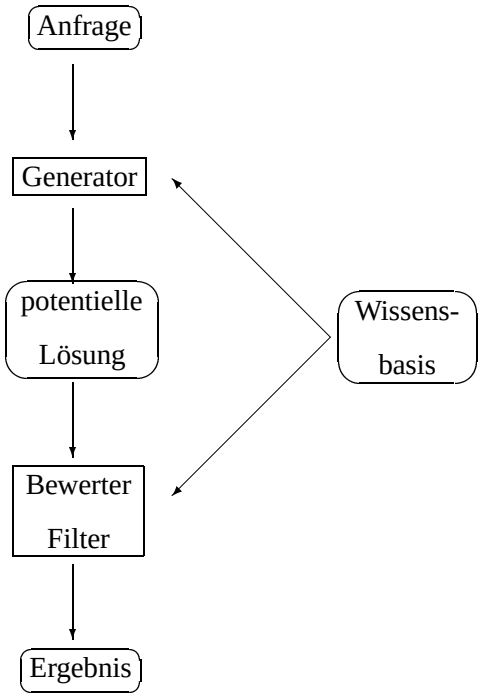

Abbildung 9: Generierung und Test, eine Grundstruktur von Expertensystemen.

Ein guter Ansatz ist *stepwise refinement*. Die Lösung wird im *top level* formuliert und schrittweise bis zum *bottom level* verfeinert. Die Strategie von *top–down* verfeinert schrittweise die Relationen. Dies sind die Grundsätze des „strukturierten Programmierens“. Dieses *Prototyping* ist auch die Stärke von Prolog. Aus wenigen Zeilen wird zuerst ein funktionierendes Programm, der Prototyp, entwickelt. Dieser wird kontinuierlich zum vollständigen System weiterentwickelt[174].

Die Vorteile, die Prolog bietet, um ein wissensbasiertes System aufzubauen, sind:

- Es existieren einheitliche Sprachkonstrukte. Es gibt nur Terme und aus Termen aufgebaute Klauseln.

[174] siehe Niklaus Wirth, [123], 1971.

- Implizite Kontrollmechanismen wie Unifikation und Backtracking sind in Prolog enthalten.
- Dynamische Änderungen der Wissensdatenbank sind möglich. In aktuellen Programmen lassen sich überflüssige Klauseln entfernen oder Zwischenergebnisse hinzufügen[175].

[175] siehe Ralf Cordes, [35], 1992, Seite 170.

5.4.7 Deklarative und prozedurale Bedeutung

Auf Kowalski[176] geht folgende Entdeckung zurück:

> „Einer der wichtigsten Gesichtspunkte des logischen Programmierens war die Erkennung der Tatsache, daß man jeden Algorithmus in zwei voneinander unabhängige Anteile zerlegen kann: in einen logischen Anteil und einen Kontrollanteil. ... Der logische Anteil besteht aus der Feststellung **was** das Problem ist, welches zu lösen ist. Der Kontrollanteil beschreibt **wie** ein Problem gelöst werden soll.“[177]

Die Unterscheidung zwischen Eingabe– und Ausgabeparameter ist in Prolog nicht einfach möglich. Als Ausgabeparameter können noch am ehesten uninstanziierte Variablen angesehen werden. Als Eingabeparameter lassen sich instanziierte Terme betrachten. Nimmt man jedoch Listen mit instanziierten und uninstanziierten Elementen, sind diese Listen sowohl Eingabe– als auch Ausgabeparameter[178]. In Prolog werden keine Algorithmen zur Abarbeitung durch Prozeduren geschrieben. Es ist durchaus möglich einen imperativen Programmierstil zu verwenden, aber dabei werden die Vorteile von Prolog aufgegeben. Im Gegensatz zu anderen Programmiersprachen können Prozedur und Algorithmus nicht synonym verwendet werden. Prolog ist nichtalgorithmisch. Oft wird „nichtalgorithmisch“ mit „nichtprozedural“ gleichgesetzt. Aber der ‚Cut‘ macht Prolog prozeduraler als es sein müßte. Im Gegensatz zu anderen Programmiersprachen kann die deklarative Bedeutung eines Programms korrekt sein und das Programm ist prozedural inkorrekt und findet keine Lösung[179]. Deshalb ist für eine Bearbeitung die Reihenfolge der Klauseln wichtig.
Für folgendes Programm findet Prolog keine Lösung, obwohl das Ergebnis $\mathtt{S} = \mathtt{a}$ bei der Anfrage $\mathtt{p(S,b)}$. offensichtlich ist:

```
1 p(X,Z) :- p(Y,Z), q(X,Y).
2 p(W,W).
3 q(a,b).
```

Durch Vertauschen der beiden Literale in der Klausel der ersten Zeile findet Prolog sofort die Lösung. Guter Programmierstil ist, ein Programm so zu strukturieren, daß einfachere Anfragen zuerst abgearbeitet werden[180]. Niemals sollten die Klauseln so aussehen:

[176] siehe Robert A. Kowalski, [72], 1979.
[177] Josef Weingärtner, [119], 1989, Seite 1.
[178] Ein Beispiel wäre $\mathtt{member(bedroht, [kuba, X, amerika])}$.
[179] siehe Ivan Bratko, [21], 1994, Seite 58.
[180] siehe Ivan Bratko, [21], 1994, Seite 60f.

```
klausel1(X):-
  klausel1(Y),
  klausel2(Y,X).
```

Die richtige Lösung ist immer:

```
klausel1(X):-
  klausel2(Y,X),
  klausel1(Y).
```

In der Modellbildung wird teilweise nach „prozeduralem“ und „deklarativem“ Wissen unterschieden. Für praktische Anwendungen ist diese Unterscheidung jedoch irrelevant[181]. Erwähnt werden muß, daß auch prozessuale, objektorientierte und funktionale Interpretationen eines Prologprogramms möglich sind[182].

[181] siehe Wolfgang Balzer, [9], 1997, Seite 118.
[182] siehe Joachim Stender, [118], 1987.

6 Soziale Simulationen

Ein Computerprogramm entspricht einer Modellklasse oder den Axiomen für eine solche. Es produziert aus einem eingegebenen Input *I* einen Output *O*. Das Schema einer Theorie $Randbedingung + Gesetz \Rightarrow Vorhersage$ wird zur Grundfigur der Computersimulation $Input + Programm \Rightarrow Output$[183]. Ein realitätsnahes Input–Outputpaar wird *korrekt* genannt. Die erwünschten Eigenschaften eines Simulationsprogramms sind Korrektheit und Vollständigkeit. Dieses Ziel kann leider meist nur graduell erreicht werden. Abhängig von den In– und Outputdaten kann man zwischen

- harten Simulationen,
- weichen Simulationen und
- modellbildenden Simulationen unterscheiden[184].

Wenn in einer Theorie präsentes Wissen in Daten, Axiomen und Definitionen aufgeschlüsselt wird, muß ein Computerprogramm Regeln für diese drei Komponenten enthalten[185]. Durch Abstriche am theoretischen Gehalt der Axiome eines Modells lassen sich die Axiome in eine logisch „gleichwertige“ Regelmenge überführen. Umgekehrt läßt sich jedes Computerprogramm durch ein Axiomensystem ersetzen[186].

> „Computerprogramme sind keine Theorien, sie entsprechen vielmehr den Axiomen für eine Theorie. Modelle entsprechen den Programmabläufen, der Modellklasse einer Theorie entspricht auf der Computerseite die Klasse aller Programmabläufe.“[187]

Soziale Simulationen basieren auf Agenten, die in einer abstrahierten Umwelt miteinander interagieren.

> „Agents are machines or beings that act in a world. We distinguish between the internal workings of an agent and the external world that affects, and is affected by, that agent. All that can be observed is the external world.“[188]

[183] siehe Wolfgang Balzer, [9], 1997, Seite 315.
[184] siehe Wolfgang Balzer, [9], 1997, Seite 317.
[185] siehe Wolfgang Balzer, [9], 1997, Seite 233.
[186] siehe Wolfgang Balzer, [9], 1997, Seite 119.
[187] siehe Wolfgang Balzer, [9], 1997, Seite 123.
[188] siehe Michael Georgeff, [53], 1984, Seite 121.

Müller[189] unterteilt die Simulationen mit Agenten in fünf Kategorien:

- reaktive Agenten (*reactive*),
- planende Agenten (*deliberativ*),
- interagierende Agenten (*interacting*),
- Schichtenansatz (*layered approach*) und
- andere.

Die Kategorie „andere“ beinhaltet *believable agents*, *softbots* und kommerzielle agentenbasierte Systeme. Für die Klassifikation dieser Agenten werden zwei Dimensionen vorgegeben:

- Hardwareagenten und
- Softwareagenten.

Eine weitere Dimension ist die Unterscheidung zwischen:

- autonomen Agenten,
- Multiagenten und
- Hilfsagenten.

Die Faktenbasis wird durch das Programm von abstrakten Simulationen meist selbständig generiert, um den Aufwand zur Sammlung einer Datenbasis zu umgehen. Die Gründe für dieses Vorgehen sind vielfältig:

- Probleme der Sammlung, Klassifikation und Interpretation großer Datenmengen können umgangen werden,
- alle Faktoren, ihre Interpretation, Ursache–Wirkungszusammenhänge sind völlig unter Kontrolle,

[189] siehe Jörg P. Müller, [91], 1999, Seite 212.

- beliebiges Experimentieren ist möglich (keine ethischen Probleme) und
- neue Verknüpfungen können „erfunden“ werden.

Das Sammeln einer Datenbasis entfällt. Da keine realen Daten vorliegen, gibt es keine lokal wirksamen Anwendungen und Implikationen. Trotzdem können verschiedene Modelle erstellt und mit der Realität verglichen werden[190]. Zwei Klassen von Computermodellen sind in den Sozialwissenschaften üblich:

- statistisches Modell und
- Simulationsmodell.

Hinsichtlich des Zeitaspektes unterscheidet man zwischen statischen und dynamischen Modellen. Tendenziell verlagern sich die Anwendungen von Computerprogrammen vom Hypothesentest zur Hypothesengenerierung. Soziale Systeme wurden zuerst auf der Basis von Differentialgleichungen untersucht. Der Vorteil gegenüber der Simulation liegt in einer vollständigen Verhaltensbeschreibung des Modells. Jedoch ist diese Methode für die Beschreibung komplexer Realitätsausschnitte ungeeignet.
Den Simulationen liegt meist eine diskret–synchrone Zeitstruktur zugrunde. Der Zeitfortschritt erfolgt in einem von außen angelegten Zeittakt.
Eine bekannte Makrosimulationen ist der *Systems Dynamics*–Ansatz[191] DYNAMO[192]. Mikrosimulationen sind wesentlich vielfältiger. Sie sind meist konkrete Einzelentwicklungen[193]. Die Mehrebenensimulation untersucht den Zusammenhang zwischen Individuen und durch sie gebildete Population. Ein Beispiel ist das Programm MIMOSE[194].
Im folgenden werden vier Simulationen vorgestellt. EMOREGUL simuliert Motivation und Kognition bei der Handlungsregulation. Axelrods Simulation ist spieltheoretisch und

[190] siehe Wolfgang Balzer, [9], 1997, Seite 314.
[191] siehe Jay Wright Forrester, [46], 1980, sowie Dennis Meadows, [85], 1972.
[192] siehe A. Pugh, [105], 1976.
[193] siehe Richard Hauser, [62], 1993.
[194] siehe Michael Möhring, [88], 1990.

soll das Auftreten neuer Akteure simulieren. SMASS ist eine sequentielle Simulation verschiedener Akteure, die miteinander interagieren. In DMASS ist diese sequentielle Simulation auf ein Transputersystem übertragen. Die Unterscheidung zwischen SMASS und DMASS basiert auf einer hardwarenahen Grundeinteilung zwischen sequentiellen und parallelen Systemen.

Verhandlungen sind ein Kommunikationsprozeß zwischen verschiedenen Akteuren, in dem unterschiedliche Ziele zu einer gemeinsamen Übereinkunft führen[195]. Die meisten Verhandlungen sind unflexibel und koordinieren nur die Ausführung von Handlungen. Soweit Ziele und „*beliefs*“ der beteiligten Akteure nicht berücksichtigt werden, sind Verhandlungen zwischen den Akteuren auch nicht notwendig.

[195] siehe Man Kit Chang, [29], 1992, Seite 32.

6.1 EMOREGUL

Das Programm EMOREGUL von Dietrich Dörner ist eine Simulation der Interaktionen von Motivation, Emotion und Kognition bei der Handlungsregulation. Die Programmiersprache ist PASCAL..
Mit dieser Simulation soll „zielgerichtetes Mehrfachhandeln in komplexem, dynamischen Umfeld“[196] erklärt werden.

> „Am Anfang ist weder Wort noch Tat, am Anfang stehen die Bedürfnisse . . . .“[197]

Bedürfnisse können adaptiv oder aversiv (man möchte etwas haben oder vermeiden) sein. Sie können von außen (Kühle an heißen Tagen) oder von innen induziert werden, wobei die inneren Bedürfnisse wiederum unterschieden werden nach:

- konsumptorisch (Hunger, Durst) und
- periodisch (Schlaf) wiederkehrend.

Wenn ein Bedürfnis einen bestimmten Schwellwert überschreitet, entsteht ein Motiv. Der Schwellwert ist variabel und abhängig von schon existierenden Motiven. Gibt es mehrere Motive, wird die Auswahl nach folgenden Kriterien getroffen:

- Wichtigkeit (Größe der Sollwertabweichung),
- Dringlichkeit (Zeitgebundenheit) und
- Fähigkeit zur Befriedigung des Bedürfnisses (Faktenwissen, Erfahrung, epistemische Kompetenz).

Das Produkt von Wichtigkeit, Dringlichkeit und Fähigkeit zur Bedürfnisbefriedigung (Erfolgserwartung) ergibt den Auswahldruck für ein Motiv. Vor dem Agieren zur Bedürfnisbefriedigung eines ausgewählten Motivs wird ein Plan erstellt. Das Modell der psychischen Prozesse des Menschen, das durch EMOREGUL dargestellt wird, ist dynamisch,

[196] Dietrich Dörner, [39], 1996, Seite 4.
[197] Dietrich Dörner, [39], 1996, Seite 4.

quantitativ und diskret (Zeit wird in Takten abgebildet). Das Nachfolgeprogramm von EMOREGUL ist PSI[198].

> „In dem PSI-Projekt wird eine Theorie zur Erklärung des Handelns von Menschen in komplexen Situationen weiterentwickelt. Die Theorie erlaubt es, das Handeln und Erleben von Menschen in komplexen Realitätsbereichen zu erklären und vorauszusagen. Dabei geht es im einzelnen um folgende Fragen:
> Wie bilden Menschen Ziele und Absichten? Wie behandeln Menschen Absichten, wie und wann planen und explorieren sie; wie entscheiden sie sich für bestimmte Aktionen? Wie bilden und verändern Menschen Gedächtnismodelle komplexer Realitäten?
> Die Theorie wird in die Form eines Computerprogramms gebracht, damit die Widerspruchsfreiheit und Vollständigkeit der Theorie geprüft werden kann und damit Schlußfolgerungen aus der Theorie gezogen werden können. Diese Schlußfolgerungen haben die Gestalt künstlich erzeugter "Handlungen"; der Computer „simuliert“ das Verhalten von menschlichen Versuchspersonen. Dieses Verhalten der „künstlichen“, computersimulierten Versuchspersonen wird mit dem von „echten“ Versuchspersonen verglichen. Durch solche Vergleiche kann die Theorie geprüft werden.“[199]

Die Bedürfnisse werden mit potentiellen Zielzuständen verknüpft. Das Wissen des Systems vergleicht erwünschte Zielzustände, geeignete Handlungsschrittte, sowie Wichtigkeit und Dringlichkeit der Absicht. Die Absicht mit der höchsten Dringlichkeit und Erfolgswahrscheinlichkeit wird zur Ausführung gebracht. Der Agent existiert in einer künstlichen Umwelt, auf die er Einfluß nehmen kann. Das System lernt mit der Zeit seine Umwelt kennen. Erfolg oder Mißerfolg werden protokolliert. Die Informationen der Datenbank stehen dem Agenten später zur Verfügung. Angereichert ist EMOREGUL mit der Möglichkeit Daten zu vergessen. Zusätzlich werden Emotionen simuliert.
Dörner räumt ein, daß komplexe Verhaltensweisen niemals Lückenlos prognostiziert werden können. Kleinste Variationen von Parametern können drastische Veränderungen im Verhalten erzeugen[200].

[198] siehe Dietrich Dörner, [40], 1999.
[199] Dietrich Dörner, [40], 1999.
[200] siehe Christina Bartl, [11], 2000.

6.2 Neue politische Akteure

In „Building New Political Actors“ entwickelt Robert Axelrod ein Modell für das Erscheinen von neuen politischen Akteuren[201]. Dieses System ist in den Programmiersprachen Pascal und Visual Basic realisiert[202].

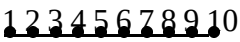

Abbildung 10: zweidimensionale Population mit 10 Akteuren.

> „In dem Maße, wie man die evolutionäre Leiter neuraler Komplexität hinaufsteigt, wird das spieltheoretisch beschreibbare Verhalten umfangreicher. Die Intelligenz der Primaten, einschließlich des Menschen, erlaubt eine Anzahl wichtiger Verbesserungen: ein komplexeres Gedächtnis, komplexere Informationsverarbeitung zur Bestimmung der nächsten Handlung als Funktion der bisherigen Interaktion, eine bessere Abschätzung der Wahrscheinlichkeit zukünftiger Interaktionen mit dem gleichen Individuum und eine bessere Fähigkeit der Unterscheidung zwischen verschiedenen Individuen. Die Fähigkeit zur Diskrimination anderer ist dabei besonders wichtig, weil sie es erlaubt, Interaktionen mit vielen Individuen durchzuführen, ohne diese Individuen alle gleich zu behandeln, so daß also die Kooperation eines Individuums belohnt und die Defektion eines anderen Individuums bestraft werden kann.“[203]

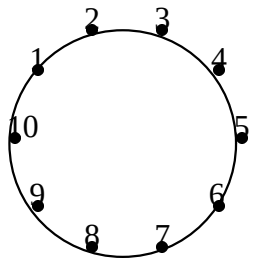

Abbildung 11: Population mit je zwei Nachbarn.

Die Population von Axelrods 10 Akteuren ist zweidimensional. Damit alle Akteure zwei Nachbarn haben, sind sie kreisförmig angeordnet.

Für diese Simulation gelten folgende Regeln:

- Jeder Akteur besitzt genau zwei Nachbarn.
- Jeder Akteur erhält ein initiales Vermögen (*wealth*: $300 \leq W_A \leq 500$).
- Eine Periode wird „Jahr“ genannt.

[201] siehe Robert Axelrod, [1], 1997, Seite 124 – 144.

[202] weitere spieltheoretische Ansätze finden sich zum Beispiel in Rainer Hegselmann, [63], 1996, Wim Liebrand, [79], 1996 und Andrej Nowak, [92], 1996.

[203] Robert Axelrod, [2], 1997, Seite 85.

- Eine typische Simulation dauert 1.000 Jahre.
- Pro Jahr werden von jedem Akteur 20 Punkte ($wealth(20)$) „geerntet“.
- Jedes Jahr werden drei Akteure aktiv. Die Auswahl der Akteure erfolgt zufällig. Der aktive Akteur *A* kann das Handeln ablehnen oder er verlangt Tribut vom Akteur *B*. Dabei muß *B* ein Nachbar sein.

Wird *B* angegriffen hat er die Wahl, Tribut zu zahlen oder zu kämpfen (*pay or else*). Entscheidet *B* zu zahlen, erfolgt ein sofortiger Transfer von 250 Punkten von *B* nach *A*. Ist das Vermögen von *B* niedriger als 250 Punkte, überstellt *B* sein gesamtes Eigentum an *A*. Entscheidet sich *B* für das Kämpfen, verliert *B* 25% von *A*s Vermögen. Ebenso verliert *A* 25% von *B*s Vermögen. Besitzt einer von beiden weniger, wird der Verlust anteilig verrechnet. Während der gesamten Simulation sind Verpflichtung und Vermögen gemeinsames Wissen aller Akteure.
Das ideale Opfer ist so schwach, daß es lieber zahlt als kämpft und gleichzeitig stark genug, um möglichst viel zahlen zu können. Der Angreifer versucht, das Produkt aus Verwundbarkeit des Ziels und möglicher Zahlung zu maximieren. Die Verwundbarkeit des Gegners *B* errechnet sich nach folgender Formel: $B = \frac{W_A - W_B}{W_A}$. Wenn keiner der Nachbarn schwächer als der Angreifer ist, wird keine Handlung ausgeführt. *B* kämpft, wenn ihm dies weniger kostet, als sofort zu zahlen.
Während der Simulation ändert sich die Verpflichtung der Akteure untereinander. Die Verpflichtung ist ein Zahlenwert zwischen 0 und 100 ($0 \leq Verpflichtung \leq 100$). Das Ansteigen und Senken des Wertes der Verpflichtung erfolgt in Schritten von 10 Punkten . Ansteigen und Senken der Werte erfolgt symmetrisch – d.h. steigt die Verpflichtung von *A* so auch die von *B*. Zu Beginn ist die Verpflichtung aller Akteure 0.
Die Verpflichtung von i an j steigt, wenn:

- i Tribut an j zahlt (Unterwürfigkeit),
- i Tribut von j erhält (Protektorat),

- i auf derselben Seite wie j kämpft (Freundschaft).

Die Verpflichtung sinkt, wenn:

- i auf der anderen Seite als j kämpft (Feindschaft).

B ist ein Nachbar von A, wenn für die Verpflichtung V aller Akteure i zwischen A und B gilt: $V_{iB} \leq V_{iA}$. Zur Veranschaulichung ist es am einfachsten, die Geschichte eines Akteurs zu beobachten. In Tabelle 9 auf Seite 74 ist dazu ein Beispiel dargestellt. Die Analyse der Geschichten erfolgt nach vier verschiedenen Kriterien:

1) Die einzelnen Akteure werden betrachtet.
2) In der globalen Sicht werden gemeinsame Cluster von Vermögenswerten und die Anzahl der Kämpfe errechnet.
3) Die Verpflichtungen der einzelnen Akteure bilden einen weiteren Untersuchungsgegenstand.
4) Tributzahlungen und Kriege können aufgelistet werden.

Ein bemerkenswertes Ergebnis der Simulation ist, daß bei jeder fünften Simulation die Werte kollabieren. Bildet sich ein starker Akteur heraus, bleibt er meist stark und entwickelt sich. Da der Starke ein lohnendes Ziel sucht, wechselt er unter den Schwachen, die dadurch schwach bleiben. Wenn der Starke neutral bleibt, kommt es zu „Bürgerkriegen“. Weiterhin ist festzustellen, daß initiale *wealth*–Werte auf Dauer keinen Einfluß haben, da hohe *wealth*-Werte einen Akteur zu einem lukrativen Ziel machen. Starke Akteure können aber durch schwache in einen „Weltkrieg“ hineingezogen werden (*imperial overstretch*). Normalerweise wird bei dieser Simulation kein stabiler Status erreicht. *Imperial overstretch* kann auch die Stärksten vernichten. Sind verschiedene, benachbarte Akteure untereinander verpflichtet, spricht Axelrod vom Entstehen neuer Akteure. Seine Kriterien für neue Akteure sind:

1) Effektive Kontrolle über Untergebene:

- Revolution (gegen den Stärksten) kann nicht beobachtet werden und
- unabhängige Außenpolitik ist die Ausnahme.

2) Gemeinsame Aktionen (Alle für Einen, Einer für Alle):

- Patronat (Schutz der Schwachen durch Stärkere),
- gemeinsame Außenpolitik.

3) Wahrnehmung der einzelnen verpflichteten Agenten durch die Umwelt als ein Akteur.

Diese Kriterien werden durch das vorgestellte System erfüllt. Somit kann wirklich vom Auftreten neuer Akteure gesprochen werden. Einsichtiger wird dies, wenn berücksichtigt wird, daß Axelrod die Geschichte der Vereinigten Staaten modellieren wollte. Dreizehn Staaten haben sich verbündet, um gegen England vorzugehen.
Axelrods spieltheoretische Simulation ist von der EdK–Gruppe unter Rudolf Schüßler[204] mit Mobilitätsannahmen erweitert worden. In dieser Simulation können Agenten ihre Nachbarn tauschen.

Jahr 9 5 wird aktiv und greift 4 an.
4 entscheidet zu kämpfen.

Jahr 10 5 greift wieder 4 an.
Geschwächt durch den vorherigen Kampf entscheidet 4 zu zahlen.
Die Verpflichtung erhöht sich um 10.

Jahr 11 5 greift, mit Hilfe von 4, 3 an.
3 zahlt. Die Verpflichtungen erhöhen sich bei 3, 4 und 5.

Jahr 14 5 greift 2 an.
2 kämpft gegen 5, 4 und 3.
Die Verpflichtungen erhöhen sich bei 3, 4 und 5.
Die Verpflichtung gegenüber 2 sinkt bei 3, 4 und 5 um 10.

Tabelle 9: Geschichte eines Akteurs Nr. 5
nach Robert Axelrod, [1], 1997, Seite 131 – 132.

[204] siehe Rudolf Schüßler, [115], 1995.

6.3 SMASS

SMASS (**S**erial **M**ulti–**A**gent **S**ystem for **S**imulation in Social Science) ist ein Multiagentensystem zur Simulation von umfassenden, sozialen Systemen wie Märkten, Organisationen und Institutionen. Die Vorteile von SMASS liegen in seiner minimalen, konzeptorientierten Implementierung von umfangreichen sozialen Systemen. Die Architektur von SMASS liefert einen Rahmen, in den der Benutzer verschiedene Regeln des Verhaltens einfügen kann, um die Auswirkungen der Regeln im Makromodell zu studieren. SMASS verwendet eine *bottom–up* Strategie. Nur die Begriffe, die für die intendierte Simulation essentiell sind, werden verwendet.

6.3.1 Hintergrund und Alternativen

Die Implementierung von SMASS basiert auf dem formalen Modell sozialer Institutionen[205]. SMASS beschreibt nur einen kleinen Teil des umfassenden Modells. Das Modell und die Implementation beinhalten nur die notwendigsten Begriffe (Konzepte, Objekte, Relationen und Funktionen) auf einem abstrakten Niveau, um die intendierten Phänomene in einer ersten Annäherung zu beschreiben. Im Modell der sozialen Institutionen wird die Beeinflussung von Personen durch andere Personen zugrundegelegt. Die Relation der Beeinflussung beinhaltet Handlungen vieler Arten. Eine soziale Institution wird als eine hierarchisch geordnete Menge von Gruppen verstanden, wobei jede Gruppe eine charakteristische Handlung durchführt. In diesem Modell ist jede individuelle Handlung durch die Gruppe und jede charakteristische Hand-

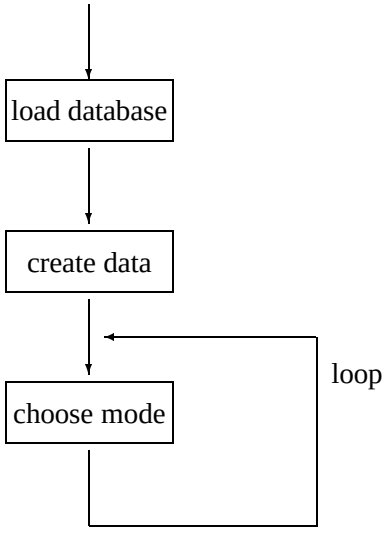

Abbildung 12: Datenbasis und Schleife in SMASS**.**

[205] siehe Wolfgang Balzer, [5], 1990 und Wolfgang Balzer, [6], 1993.

lung der Gruppe determiniert.

Ein grundsätzliches Problem der Sozialwissenschaften ist der Umgang mit Handlungen und Handlungstypen[206]. Die Handlungstypen der Alltagssprache wurden noch nicht unter soziologischer Perspektive erfaßt, so daß sie für ein Simulationsprogramm verfügbar wären.

SMASS behandelt dieses Problem wie folgt:

1) Jedem Handelnden werden verschiedene Arten des Verhaltens zugewiesen, die während der Durchführung des Programms zu jeder Zeit zufällig oder abhängig von anderen Kriterien (z.B. Charakter) gewählt werden. Beispiele sind: ausruhen, konsumieren, arbeiten oder jemanden beeinflussen.

2) Für einige dieser Handlungen wird ein numerisches Repräsentationsformat festgelegt.

Handlungen werden durch Handlungstypen repräsentiert. Jede konkrete Handlung eines Typs wird durch eine numerische Repräsentation von entsprechendem Format spezifiziert. Ein Beispiel sei ‚beeinflussen' als Zahl 3 in folgender Liste $[A, B, 3, IA, OA, IB, OB]$. Die beiden Personen sind *A* und *B*, und *A* beeinflußt *B*. IA, OA, IB, OB repräsentieren den *Input* und *Output* beider Personen während der Handlung. Wenn A zum Beispiel B anordnet, für ihn eine Arbeit zu erledigen, ist As *Input* die Äußerung der Anweisung, während As *Output* der Wert der Arbeit sein wird, die B tut. Andersherum: Bs *Input* ist die Anstrengung, die Arbeit zu tun, während Bs *Output* Null ist. Handlungen dieser Art werden von jeder Person zu jeder Zeit zufällig oder abhängig von Regeln (abhängig von Charakter und sozialer Position) ausgewählt.

Sequenzen von Interaktionen des individuellen Verhaltens können abhängig von im System vorhanden Parametern generiert werden:

- körperliche Kraft,

[206] siehe Thomas Ballmer, [4], 1986.

- Wohlstand und
- soziales Kapital.

Ein weiteres Kriterium ist Nachbarschaft. Im gegenwärtigen System werden Entscheidungen herbeigeführt, indem die erste zufällige Möglichkeit akzeptiert wird, welche die gegebenen Kriterien erfüllt. Dies vereinfacht das Programm, trotzdem können die Einflüsse der Regeln des Verhaltens und der Einfluß von Parametern auf die interagierenden Personen studiert werden. SMASS konzentriert sich weder an aktuellen Verfahren des Verhaltens noch an verschiedenen persönlichen Entscheidungen, die solche Verfahren generieren. Trotzdem sind die essentiellen Erscheinungsformen von Makrophänomenen der sozialen Institutionen beinhaltet.
In SMASS sind die Regeln des Verhaltens einfach. Weder Ziele noch Intentionen werden für ihre Formulierung gefordert. Die gegenwärtige Version verwendet eine einfache Berechnung der Vorlieben der Akteure in numerischen Termen, aber selbst diese sind nicht wirklich essentiell.
Verwunderlich ist, daß so einfache Regeln ein interessantes Systemverhalten hervorbringen. Ebenso würde man erwarten, daß Objekte, die eine realistische Beschreibung menschlichen Verhaltens beinhalten, Intentionen verwenden müssen. In der Theorie über soziale Systeme sind Intentionen nicht so zentral, wie man erwarten würde. In sozialen Interaktionen werden die Menschen durch Gewohnheiten, die sie durch Erziehung oder Imitation bestehender Verhaltensweisen erlernen, geleitet.
Ziele und Intentionen haben Einfluß auf Handlungen. Aber ist es nötig, sie beim Modellieren und Simulieren von sozialen Systemen zu berücksichtigen? Ein großer Teil der sozialen Verhaltensweisen ist weniger zielbestimmt, als durch bestehende Handlungstheorien vorgespielt wird.
Dies wird klarer beim Blick auf Multiagentensysteme für technische Anwendungen. In diesen Systemen hat jeder Agent Rezeptoren und kann ankommende Informationen bearbeiten. Wenn der Agent Ziele hat oder während der Laufzeit neue Ziele entwickelt, sind

diese immer extern vom Systementwickler initiiert oder von solchen externen Zielen abgeleitet.
Ein Roboter, der Müll sammelt, hat dieses Ziel vom Systementwickler eingegeben bekommen. Dieses Ziel hat der Roboter nicht durch Sozialisation oder Lernprozesse erlangt. Ein solches Ziel kann in einem sozialen Wesen entstehen. Der Prozeß der Sozialisation ist lang, kompliziert und beinhaltet differenzierte soziale Fähigkeiten, welche die eines Computersystems noch bei weitem übersteigen.
Die Interaktion der Roboter mit der materiellen Welt – mit Objekten und Ereignissen – wird meist über Ziele gesteuert. Andere Roboter werden primär nur in ihrer materiellen Natur und den materiellen Konsequenzen ihres Verhaltens wahrgenommen. In den meisten Multiagentensystemen findet der Hauptteil der Interaktionen zwischen Agent und seiner materiellen Umgebung statt. In der sozialen Sphäre sind einige Verhaltensformen nicht mit materiellen Begriffen darzustellen.
Die meisten Multiagentensysteme sind technische Anwendungen. Diese Systeme unterscheiden sich von sozialen Systemen in einem entscheidenden Punkt: dem Ziel. In einer technischen Anwendung existieren immer einige Parameter oder globale Quantifizierungen des Systems, welche die Anwendung maximieren. In sozialen Systemen und ihrer Simulation gibt es kein endgültiges, identifizierbares Ziel, das erreicht werden soll. Um solch ein Ziel einzurichten, müßte man etwas wie das Ziel und die Bedeutung des Lebens definieren, was unsere Kapazitäten übersteigt. Soziale Systeme haben keine identifizierbaren Ziele und keine globale Quantität, die maximiert werden.

6.3.2 Ursprünge von SMASS

SMASS ist in Prolog implementiert. Hier wird nur der Teil des Programms beschrieben, der soziale Relevanz besitzt. Die Module des *In–* und *Outputs* sowie die Art und Weise, in der das System statistisch vorgeht, werden nicht beschrieben.
In SMASS werden die in Tabelle 10 dargestellten Abkürzungen zur Darstellung ange-

wendet. Die Zahlen der Parameter können variiert werden, bleiben aber während der

Abkürzung	Bedeutung
AS	Anzahl der Handelnden im System
MM	Anzahl der Handlungsarten (die gleiche für jeden Handlungsträger)
SS	Definitionsbereich der physikalischen Stärke des Agents
WW	Definitionsbereich des Wohlstandes des Agents
SC	Definitionsbereich des sozialen Kapitals des Agents
DD	Definitionsbereich der Entfernung zwischen den Agenten
CE	Anzahl der Charakterausdrücke
EE	existentielles Minimum des Agenten (das gleiche für jeden Handlungsträger)
DI	Definitionsbereich des *Inputs* in der Powerrelation
DO	Definitionsbereich des *Outputs* in der Powerrelation

Tabelle 10: Abkürzungen in SMASS, nach Wolfgang Balzer, [7], 1996, Seite 5.

Simulation fixiert. Mindestens zwei Handlungsträger müssen vorhanden sein. Minimal existiert eine Handlungsart. Jede Handlungsart ist als eine Familie von Handlungstypen denkbar. Ein Handlungstyp ist im System gegenwärtig, wenn eine dazugehörende Regel des Verhaltens spezifiziert ist. Wenn ein Agent in einem definierten Modus ist, wird er einen Handlungtyp wählen, der zu diesem Modus paßt. Jeder Agent ist mit drei Parametern ausgestattet: *SA, WA, CA*. Diese Parameter beschreiben die physikalische Stärke *SA* von *A*, seinen Reichtum *WA* und sein soziales Kapital *CA*. Diese können keine willkürlichen Werte annehmen, sondern sind aus den Definitionsbereichen *SS, WW*, *SC* entnommen. Die Stärke des Agents ist invariant. Die beiden anderen Werte können sich ändern. Dafür hat der Agent zwei Konten. Die soziale Stärke eines Agenten ist die Summe der Stärke seiner Nachbarn. Zu Beginn wird der Definitionsbereich des Abstands benötigt, um Nachbarschaft zu kreieren. SMASS kreiert für zwei Agenten

eine Distanz, so daß die Axiome der Metrik des Raumes erfüllt sind. Diese Distanzen können verwendet werden, um Nachbarschaft verschieden zu definieren. Die Distanz zwischen zwei Handelnden muß aus dem Definitionsbereich *DD* entnommen werden ($d : AS \times AS \longrightarrow \mathbb{R}\ d(N,A) \leq E$). Auch zu Beginn kreiert SMASS Charaktere. Wenn A ein Handlungsträger ist, wird sein Charakter durch eine Liste $[A, C_1, \ldots, C_{MM}]$ beschrieben, wobei für jeden Handlungsmodus M eine Zahl C_M den Ausdruck von A's Charakter zum Modus M repräsentiert. Etwas unrealistisch gibt es für jeden Modus die gleiche Anzahl von Ausdrücken. Die Ausdrücke werden durch die Zahl $1, \ldots, CE$ repräsentiert. Wenn zum Beispiel der Modus für Konsumieren M ist, zeigt der Wert von $C_M (1 \leq C_M \leq CE)$ A's Neigung an, konsumierenden Aktivitäten nachzugehen. Eine asketische Person wird den Wert 1 $(C_M = 1)$ haben. Wenn der Wert C_M den Wert von C_E hat, bedeutet es, daß diese Person sehr gern konsumiert. Die Charaktere werden zufällig vergeben. Für jeden Handlungsmodus wird von SMASS eine Liste der Gewichte $[W_1, \ldots, W_{CE}]$ erstellt[207]. Des weiteren wird das existentielle Minimum für die Formulierung der Regeln der Interaktionen verfügbar gemacht. Unter das existentielle Minimum darf der Reichtum nicht absinken. Im Fall einer Ausbeutung durch Gewalt, in der die Person unter das Minimum absinkt, stirbt die Person und die Ausbeutung scheitert. In einem solchen Fall nutzt der Ausbeuter das existentielle Minimum, um auszuloten, wie weit er gehen kann, ohne daß die andere Person die Szene verläßt. Die Definitionsbereiche des *Inputs* und *Outputs* der Powerrelation werden für die zufällige Kreation der Handlungen verwendet, die von der Powerrelation benötigt werden. Eine Handlung besteht aus vier Zahlen: $[IA, OA, IB, OB]$, die den *Input* und *Output* der Handlungsträger A und B in einer einzelnen Aktion verkörpern. Die Beschränkung dieser Zahlen auf gewisse Definitionsbereiche schränkt die Möglichkeit der Handlungen, die gewählt oder durchgeführt werden können, ein.

[207] Die Daten der Gewichte unterliegen einer diskreten Verteilung. Der Charakter ist in jedem Agent diskret verteilt.

6.3.3 Behandlung der Zeit

SMASS ist ein sequentielles Programm und mit dem Problem der Verwaltung von Zeit konfrontiert. Diese Schwierigkeit wird auf die einfachste Weise gelöst. In einem Simulationslauf erfordert das System eine Anzahl TT von Perioden oder Zeitpunkten. Es arbeitet exakt TT Perioden und in jeder Periode hat ein Agent genau eine Möglichkeit zu handeln. Dies ist zugegebenermaßen eine idealisierte Herangehensweise. Wie lang die Dauer einer Periode in einem realen System auch sein mag, kann nicht ausgeschlossen werden, daß Personen mehrere Handlungen in einer Zeitperiode ausführen. In der realen Welt sind die Handlungstypen nicht disjunkt.
Beispiele dafür sind: Eine Person konsumiert (Abendessen) und vermehrt zur gleichen Zeit das soziale Kapital (Abendessen mit einem einflußreichen Freund) oder eine Person erhält zwei verschiedene Meldungen gleichzeitig. In SMASS sind solche Zusammentreffen über zwei oder mehrere Perioden verteilt. Diese Einschränkung wird zugunsten der Einfachheit in Kauf genommen, da jeder realistische Versuch zu Schwierigkeiten der Zeitlogik führt.
Die Synchronisation der Handlungen, zum Beispiel ‚Austausch', wird in SMASS durch die Referenz auf die Periodenzahl (T) ausgeführt. Die beiden Konten (Reichtum und soziales Kapital) jedes Agenten sind zeitabhängig und ihr Stand wird gespeichert.

6.3.4 Schlußfolgerungen

Ein Vorteil von SMASS ist, daß individuelle Überzeugungen und Intentionen völlig fehlen. Trotzdem können interessante soziale Phänomene in SMASS dargestellt werden[208]. Selbstverständlich haben Personen in der Realität Überzeugungen und Intentionen und verwenden diese zur Koordination von Handlungen. Trotzdem kann das Fehlen von Intentionen in SMASS gut begründet werden. In sozialen Systemen sind Handlungen überwiegend Interaktionen zwischen Personen. In anderen Systemen können Interak-

[208] Beispiele sind Märkte, Organisationen und Institutionen.

tionen durchaus zwischen Materiepartikeln oder nichtmenschlichen Entitäten stattfinden. Im Kontext von sozialen Interaktionen haben Überzeugungen und Ziele einen anderen Stellenwert als in nicht sozialen Kontexten. Soweit die Überzeugungen andere Personen betreffen, sind sie schwammiger und veränderbarer als die Überzeugungen über nicht menschliche Dinge. Im sozialen Rahmen sind Überzeugungen und Ziele auch wesentlich generalisierter und abstrakter[209]. Aus diesen abstrakten Überzeugungen und Zielen entwickeln sich Regeln des Verhaltens, wie ‚Nutzenmaximierung', ‚Verträge einhalten' und ‚andere ausbeuten, wenn möglich'.
Deswegen werden Überzeugungen und Ziele wenig verwendet und sind im alltäglichen sozialen Kontext weniger wichtig. Im Zusammenhang mit SMASS werden zwei vorläufige Thesen aufgestellt:

1) Überzeugungen und Intentionen sind weniger wichtig für die Modellierung und das Verständnis sozialer Systeme als dies von der Philosophie der Handlungen und durch BDI Architekturen dargestellt wird.

2) Für die Simulation sozialer Phänomene sind nur zwei Eigenschaften der Multiagentensysteme essentiell:

   (a) In Multiagentensystemen muß es den Agenten möglich sein, Informationen auszusenden und zu empfangen, in welcher elementaren Form auch immer.
   (b) Es muß Regeln geben, die das Verhalten der Agenten leiten. Sie können für jeden Agenten verschieden sein. Auch die Geschichte des Systems oder vorhergehende Interaktionen können sie beeinflussen.

Das Bestehen von Überzeugungen, Zielen und Intentionen ist für diese beiden Thesen keine Vorrausetzung.

[209] Beispiele dafür sind: „reich werden" und „andere Personen verletzen". Abstrakte Ziele beinhalten nicht alltägliche Handlungen

6.4 DMASS

DMASS[210] ist ein verteiltes Multiagentensystem für die Simulation in Sozialwissenschaften. Es läuft unter BRAIN AID PROLOG auf Transputer-Systemen. Die intendierten Anwendungen des Programms sind qualitative Simulationen von sozialen Systemen. In DMASS wird jeder Agent des Systems durch einen Transputer repräsentiert.
Existierende qualitative, soziale Simulationen werden hauptsächlich in zellulärer Automatenumgebung durchgeführt und sind auf sequentiellen Umgebungen programmiert. In DMASS können die existierenden Simulationen ohne Aufwand reproduziert werden. DMASS ist aber flexibler und eleganter als die zellulären Automaten.
Ein weiterer Vorteil gegenüber sequentiellen Systemen ist die Behandlung der Zeit in DMASS. DMASS ist nicht nur einfacher zu programmieren, sondern vermeidet auch Schnittstellen, die in sequentiellen Programmen unvermeidbar sind.
DMASS wurde durch die Anwendung einer *bottom–up* Strategie entwickelt. Nur die Begriffe, die erwiesenermaßen essentiell für die intendierte Simulation sind, werden eingeführt. Die Eigenschaften einer BDI Architektur[211] fehlen fast völlig. Der Vorteil von DMASS ist ein minimaler, reduzierter, konzeptualer Apparat, der mit umfassenden sozialen Phänomenen umgeht. Seine Architektur zielt wesentlich auf einen flexiblen Rahmen, in dem der Benutzer variable ‚Regeln des Verhaltens' einfügt. Die ‚Regeln des Verhaltens' liegen in Form von menschlichem Verhalten und Interaktionen sowie verschiedenen Arten von Botschaften vor. Dabei werden die Makroeffekte auf diese Mikroeinzelheiten studiert.

[210] siehe Wolfgang Balzer, [10], 1996.
[211] zur BDI Architektur siehe Anand Rao, [107], 1991.

7 KRIS

Das Programm beschreibt die Interaktion von Staaten. Alle beteiligten Staaten befinden sich sich in einem Krisenzustand[212]. Die beteiligten Staaten müssen nach gewissen Regeln in dieser Krisenumgebung handeln. In jeder Runde steht ein Akteur vor dem Problem eine geeignete Handlung auszuwählen. Die Akteure müssen nicht abwechselnd an die Reihe kommen. Sie können auch mehrmals hintereinander aufgerufen werden. Findet ein Akteur keine geeignete Handlung kann er mit *skip* die Runde beenden.

7.1 Idee

Als Datengrundlage benutzt die Simulation nur eine Liste von Handlungen. Auf diese werden Regeln angewendet. Die Präverenz der einzelnen Akteure für bestimmte Regeln wird Charakter genannt. Mit diesen wenigen Elementen läßt sich eine komplexe Simulation aufbauen, die auch interessant Ergebnisse liefert.

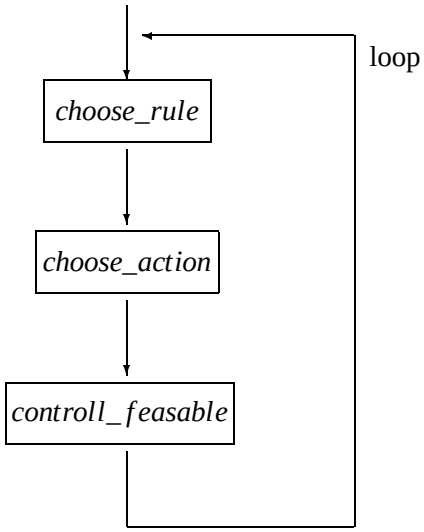

Abbildung 13: Auswahl der Handlungen.

7.1.1 Handlungstypen

Als Datengrundlage für die Simulation dienen die Handlungen aus Michael Brechers Artikel „Toward a Theory of international Crisis Behavior“[213]. Aus dieser kompakten Form werden dreißig Handlungstypen ausgewählt und vorerst wie in diesem Artikel in Gruppen eingeteilt[214]. Die spezifischen Handlungen in

[212] zur Entstehung dieses Krisenzustandes siehe Wolfgang Balzer, [8], 1996 und Jörg Sander, [111], 1993.
[213] siehe Michael Brecher, [22], 1977.
[214] siehe Tabellen 12 bis 17, Seite XIX bis XXI im Anhang.
Weitere Daten zu Konflikten finden sich bei:
Michael Brecher, [25], 1988
Klaus Jürgen Gantzel, [51], 1992

Tabelle 12 im Anhang auf Seite XIX sind illustrativ. Die Menge der Handlungen kann erheblich ausgeweitet werden. Durchgeführt werden die Handlungen von einzelnen Akteuren. Wie dieser Akteur beschaffen ist, bleibt unberücksichtigt. Der Agent kann aus einem oder mehreren Staaten bestehen, die als Block oder Koalition gemeinsam eine Handlung durchführen[215]. Die Handlungsauswahl ist eingeschränkt. Eine Datenbasis enthält alle 32 möglichen Handlungen in folgender Form:

```
act(Handlung,Staerkegrad1,Maximalstaerke,Nummer).
```

Eine Handlung besteht aus dem Namen der Handlung, dem Stärkegrad eins, dem maximalen Stärkegrad und einem Zähler.
Beispiele sind:

```
act(angriffsdrohung,1,5,1).
act(beschuldigung,1,3,2).
act(vergeltung,1,5,3).
```

Die Namen der Handlungen sind nur Abkürzungen der von Brecher definierten Begriffe. So bedeutet `handel` ein Aufkündigen einer Handelsvereinbarung. Die Bedeutung der Abkürzungen unterscheidet sich oft von einer intuitiven Zuordnung. Für das Verständnis sollten immer die Tabellen 12 und 13 auf Seite XIX und Seite XX zu Rate gezogen werden. Je nach verwendeter Regel werden diese Handlungen Gruppen zugeordnet. Da nach Ballmer das Erweitern einer solchen Liste von Verben zu neuen Gruppenbildungen führen kann[216], wird dem Programm keine allgemeine Gruppenbildung vorgeschrieben. Die Ordnungen der Handlungen sind in den Regeln implementiert. Diese Regeln können vom Benutzer bei Bedarf geändert werden. Gruppenbildungen der Handlungen dürfen sich in

Frank R. Pfetsch, [103], 1991.

[215] Zum Zusammenwachsen von Blöcken siehe Robert Axelrod, [1], 1997, Seite 124 – 144. Eine Beschreibung dieses Artikels findet sich in dieser Arbeit in Abschnitt 6.2 ab Seite 71.

[216] siehe Abschnitt 3.1 auf Seite 25.

verschiedenen Regeln auch widersprechen. Je nach Regel kann ein Verb einer anderen Gruppe zugeordnet werden.

Zuerst wird vom Programm eine Regel ausgewählt. Dann erfolgt die Auswahl der zu dieser Regel gehörenden Ordnung über die Handlungen. Ein konkretes Beispiel macht diesen Sachverhalt klarer. Die Handlung „angriffsdrohung“ kann in einer bestimmten Regel den „verbalen“ Handlungen zugeordnet werden. Eine andere Regel verwendet die Handlung „angriffsdrohung“ in der Gruppe „gewaltlos militärisch“. Die Gruppenbildung der Handlungen unterliegt stark den subjektiven Bewertungen des Programmbenutzers.

So kann durchaus die „Grenzüberschreitung mit begrenzten Kräften“ des Stärkegrades 3 stärker eskalierend als eine „Invasion des Luftraums“ des Stärkegrades 1 bewertet werden.

Bei der Ausführung von Handlungen muß zwischen Bedingungen der Ausführung, den Folgen und den Kontexten von Handlungen unterschieden werden.

> „Bedingungen sind Voraussetzungen oder charakteristische Umstände, die erfüllt sein müssen, damit eine Handlung ausgeführt werden kann. Die Folgen sind erwünschte oder unbeabsichtigte Nebeneffekte. Der Kontext definiert den Rahmen einer Handlung.“[217]

Im Programm werden die Bedingungen und Kontexte durch Regeln verwaltet. Die Folge einer Handlung ist das Schreiben dieser Handlung in eine Datenbank. Bei der Auswahl einer neuen Handlung hat diese „Geschichte“ natürlich Einfluß auf die Auswahlmöglichkeiten.

Um die Schwierigkeiten einer allgemeinen Bewertung von Handlungen zu umgehen, wird die Ordnung und Gruppierung der Handlungen also in die Regeln verlagert. Um das Programm flexibel zu handhaben, stehen verschiedene Regelgruppen zur Verfügung. Die Frage ist bei allen Regeln: Welche Handlung ist eine geeignete Antwort auf eine vorhergehende Handlung? Regeln beschreiben das reale Leben niemals vollständig. Aber um Probleme zu lösen, muß man sich nur auf die Regeln konzentrieren, die für das Problem

[217] Frank H. Piekara, [104], 1985 Seite 2.

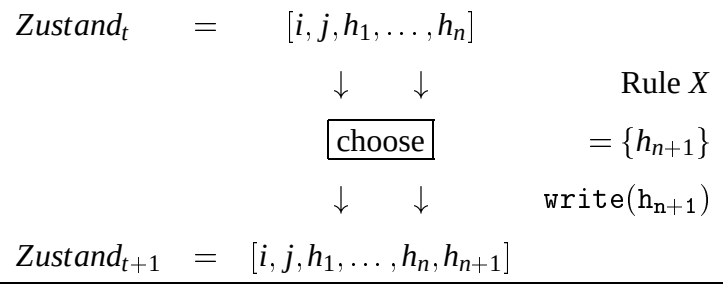

Abbildung 14: Auswahl der Handlung: Die Handlungen eines Akteurs *i* oder *j* wird nach der Auswahl durch eine Regel *X* in die Datenbasis geschrieben.

relevant sind[218]. Grenzfälle der verwendeten Regeln sind:

- antworte immer mit der stärksten Handlung (z. B. Atomschlag),
- antworte immer mit der gleichen Handlung oder
- antworte immer mit der schwächsten Handlung (z. B. diplomatische Anfrage).

Berücksichtigt werden die Folgen von Handlungen. Hat der Gegner erheblich eskaliert und ist stärker, ist die Handlung `aufgeben` naheliegend. Das Prädikat `aufgeben` läßt mehrere Interpretationsmöglichkeiten zu. Im schlimmsten Fall löst sich der betroffene Staat auf, da seine Bewohner ausgelöscht werden (*genocide*). Eine harmlosere Lösung ist ein Umsturz im betroffenen Land. Durch einen erzwungenen Regierungswechsel wird die Konfliktsituation bereinigt.
Vor jeder Runde müssen sich die Agenten entscheiden eskalierend oder deeskalierend zu handeln. Als Entscheidungsparameter dienen die ausgewählten Regeln. Zusätzlich können Daten der „*History*“ ausgewertet werden. Die Agenten verfügen über ein gemeinsames Gedächtnis. Alle Daten sind allen Akteuren bekannt.

[218] siehe William F. Clocksin, [33], 1994, Seite 2.

Bei der Regelauswahl stehen verschiedene Strategien zur Disposition. Während freundliche Strategien auf Eskalation verzichten, zeichnen sich unfreundliche durch eben diese aus. Jedoch ist die Provozierbarkeit oder Vergeltungsneigung davon unabhängig. Freundliche Strategien haben defensive Komponenten, besitzen aber keine offensiven Bestandteile. Die Strategien können weiter unterteilt werden nach:

- Musterorientierung (Regeln beziehen sich nicht auf Geschichte),
- Geschichtsorientierung (die eigene Geschichte wird betrachtet) und
- Milieuorientierung (die Geschichte des Gegners wird betrachtet).

Die Regeln können in verschiedene Gruppen eingeteilt werden:

- Regeln, die Handlungen **generieren**,
  Beispiele: existiert Stärkegrad 3, so auch 1.
  Regel: Wenn Handlung mit Stärkegrad X ($X > 0$) existiert, so auch die Handlung mit dem Stärkegrad $X - 1$.
- Regeln, die Handlungsnamen **gleichsetzen**:
  Beispiel: Bombardement Stärke 5 = Atomschlag
- Regeln, die mögliche **Handlungsfolgen** betreffen:
  Kontrolle der Datenbasis
  Beispiel: Angriffsdrohung – Angriff (temporär)
- Regeln, welche die **Ausführung** betreffen:
  Beispiel: wird die Handlung der Stärke X ($X > 0$) ausgeführt, so auch Handlung der Stärke $X - 1$.

7.1.2 Charaktere

Die Regelgruppen für einen Akteur entsprechen Charakterprofilen. Die Auswahl der Regel spiegelt den Charakter und die subjektive Einschätzung des Akteurs wider. Die

Charaktere bestehen aus Listen, welche die Präferenz eines Akteurs für eine Regel widerspiegeln. Diese Listen werden mit der Stärke und der Information, ob der Akteur ein Fanatiker ist, angereichert.
Ist „Aufgeben“ nicht im Charakterprofil enthalten, handelt es sich um eine fanatische Gruppe.
Es können drei verschiedene Arten von politischen Systemen unterschieden werden:

- Demokratien,
- zivile autoritäre Regime und
- Militärregime[219].

Diese Systeme unterscheiden sich erheblich in verschiedenen Akteursvariablen wie Krisenauslöser, Krisenmanagementtechnik und Art der Gewaltanwendung. Je autoritärer ein Regime ist, um so leichter neigt dieses Regime zur Gewaltanwendung. Die Erklärung für dieses Verhalten ist, daß sich autoritäre Regime oder gar Militärdiktaturen leichter gewaltsamer Mittel bedienen können[220]. Bei Militärregimen neigen 63% zur Eskalation. Im Gegensatz dazu liegt der Prozentwert von Demokratien nur bei 37% .

> „The leaders of military regimes are the most likley to rely on violence in the eskalation phase, whatever the nature of the initial catalyst. Violence is normal behavior for the military in power.“[221]

Ein weiteres Kriterium für die Eskalationsneigung von Staaten ist ihre interne Instabilität.

> „The more intense the internal turmoil, the greater the disposition to war.“[222]

Die Charaktere sind also abhängig von Kultur und historischer Struktur[223].
Militärische Stärke und wirtschaftliche Kraft sind die wichtigsten Grundlagen nationaler

[219] siehe dazu Michael Brecher, [26], 1998, Seite 196. Von 627 Krisenakteuren sind 43% Demokratien, 38% zivile autoritäre Regime und 19% Militärregime.
[220] siehe Michael Brecher, [26], 1998, Seite 197.
[221] Michael Brecher, [24], 1993, Seite 147.
[222] Michael Brecher, [24], 1993, Seite 149.
[223] zu weiteren Attributen der Akteure siehe Tabelle 16 auf Seite XXI und Tabelle 17 auf Seite XXI.

Macht[224]. Die Stärke eines Akteurs ist im Programm eine abstrakte Größe. Wie sich diese Stärke berechnet, bleibt weitgehend unberücksichtigt. Erwähnt seien nur die Komponenten, die Grundlagen nationaler Macht sind:

- geistige und emotionale Grundlagen (Religion, Erfahrung, Moral, Ideologie, usw.),
- legale Grundlagen (Anerkennung durch andere Organisationen),
- Organisation, Kommunikationsfähigkeit, Information,
- Bevölkerung, Arbeitskräfte, „Menschenmaterial“,
- Rohstoffe, natürliche Ressourcen,
- Wirtschaftsstärke, Industrialisierung, Stahl– und Energieproduktion, Handel, Urbanisierung, Bruttosozialprodukt,
- Territorium, geographische Lage und
- militärische Mittel (Soldaten, Waffen)[225].

Diese Komponenten allein reichen jedoch nicht aus; sie müssen auch wirksam gemacht werden können:

> „Das Verhältnis zwischen Machtgrundlagen und den Mitteln ihrer Mobilisierung ist komplex. Es ist keinesfalls so, daß derjenige, der die meisten Ressourcen und die besten Mittel besitzt, diese auch nach seinen Wünschen wirksam werden lassen kann.“[226]

Mehrere Autoren benutzen eine Vielzahl verschiedener Indikatoren, um die unterschiedliche Stärke der Staaten zu berechnen. Clifford German[227] verrechnet

- nationale Wirtschaftskraft,
- Land,

[224] siehe Frank Pfetsch, [100], 1995, Seite 80.
[225] siehe Frank Pfetsch, [100], 1995, Seite 88.
[226] Frank Pfetsch, [100], 1995, Seite 88.
[227] siehe Clifford German, [54], 1960.

- Bevölkerung und
- militärische Stärke.

Die absolute Größe jedes Lands wird durch zusätzliche Merkmale modifiziert. Mathematisch geschieht dies durch dividieren der empirisch erhaltenen Werte mit subjektiv festgelegten Zahlen.
Wilhelm Fucks[228] bestimmt die Stärke eines Staates (S) als Produkt von Stahl und Energie (P) und der Kubikwurzel der Bevölkerung (B):

$$M = P \times \sqrt[3]{B}$$
$$P = Produktion$$
$$B = Bevölkerung$$

Im hier verwendeten Programm ist die nationale Stärke (S) einfach ein Zahlenwert zwischen eins und hundert($1 \leq S \leq 100$) ohne Berücksichtigung der Entstehung dieses Wertes.

[228] siehe Wilhelm Fucks, [47], 1967.

7.2 Realisierung

7.2.1 Programmstart

Die Simulation ist in Prolog programmiert. Der Aufruf von Prolog erfolgt durch Eingabe von p*l* in der Kommandozeile. Bei einem erweiterten Programm reicht der Speicher, der Prolog beim Standardaufruf zugeteilt wird, nicht aus. In diesen Fällen muß der Aufruf mit `pl -A40m -T40m -G40m -L40m` durchgeführt werden. Die Parameter des Aufrufs weisen jedem Speicherbereich statt vier Megabyte volle vierzig Megabyte zu[229].

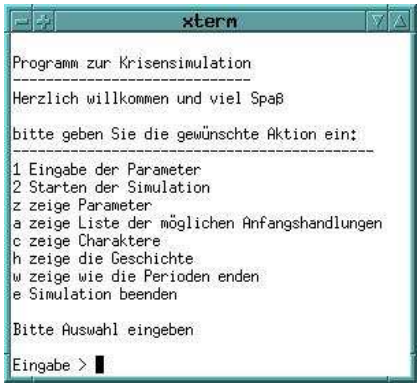

Abbildung 15: Startbildschirm des Menüprogramms.

- Der nächste Schritt zum Starten erfolgt in der Auswahl der Datei, die zu konsultieren ist. `consult('<Datei>')` kann auf die Dateien prog.prolog, monitor.prolog und stat.prolog angewendet werden. Das Grundprogramm wird mit `start/0` angestoßen.

- Die zweite Datei ist ein Menüprogramm zur Steuerung des einfachen Grundprogrammes. Gestartet wird das Programm mit `menu/0`. Dieses elegante Menüschema (*menueshell*) wurde von St. Greenwood[230] entwickelt. Prolog interpretiert seine Texte selbst, liest also nicht eine Anweisung nach der anderen. Der jeweils zu

[229] Um dies zu ermöglichen, muß der Rechner mit genügend RAM–Speicher ausgerüstet sein.
[230] siehe St. Greenwood, [60], 1984/85.

interpretierende Term aktiviert ein Prädikat mit entsprechendem Funktor und passender Stelligkeit. Interaktiv einstellbar sind die Parameter wie Runs und Perioden des Programms. Die Datenbanken werden mit dem Betriebssystembefehl `less` angezeigt. Eine einzelne Simulation kann gestartet werden. Bei Bedarf kann dieses Menüprogramm einfach erweitert werden. Auf Dauer erscheint jedoch ein grafisches *frontend* mit XPCE sinnvoller[231].

- Das letzte der drei Programme ist das Statistikprogramm. Das Erzeugen von Ergebnislisten wird mit `stat/0` eingeleitet. Dieses Programm verwendet eine Auswahl von Charakteren, für die alle ein vollständiger Programmlauf mit 20 Runs und 1000 möglichen Perioden erstellt wird[232].
  Die Ergebnislisten sind sehr umfangreich[233].

Das Kernprogramm beginnt mit einigen statistischen Funktionen. Für die Abarbeitung ist allein der Aufruf `begin/0` in Zeile 18 relevant. Dieses Prädikat leitet die Abarbeitung ein.

```
1  start :-
2    get_time(X),
3    statistics(cputime,CPUOLDTIME),
4    statistics(inferences,Inferences),
5    protocol('protokoll.txt'),
6    style_check([-singleton]),
7    ['pred.prolog', 'para.prolog', 'actions.prolog',
8    'feasable.prolog', 'rules.prolog', 'flags.prolog'],
9    consult_facts,
10   del('history.prolog'),
11   del('ende.prolog'),
12   asserta(run(1)),
13   use_old_data(X),
14   (X = no,
15   initsialisierung(1);
```

[231] siehe Abschnitt 5.3 auf Seite 48.
[232] siehe Abschnitt 7.2.7 ab Seite 105.
[233] Die Größe der Dateien beträgt ca. 800KB.

```
16  X = yes,
17  get_old_data(1)),
18  begin,
19  statistik(CPUOLDTIME, Inferences),
20  get_time(Y),
21  write_time(X, Y),
22  put(7),
23  protocolwrite,
24  noprotocol.
```

Die Prädikate `statistics/2` in Zeile 3 und 4 stellen den momentanen Status des Computersystems fest. Die Laufzeit der CPU (**C**entral **P**rocessing **U**nit) und die Anzahl der von Prolog bearbeiteten Ableitungen (*inferences*) werden festgehalten. Ausgewertet werden die gewonnenen Daten durch `statistik/2` am Ende der Simulation. Die Werte werden erneut ermittelt. Von den gewonnen Werten wird der Anfangswert subtrahiert, um den Wert, der sich auf diesen Programmlauf bezieht, zu gewinnen. Genauso wird mit der absoluten Zeit in Zeile 2, 20 und 21 verfahren.

Mit `protocol/1` in Zeile 5 wird eine Protokolldatei geöffnet, in die alle Bildschirmausgaben geschrieben werden. Zeile 24 beendet die Protokollierung und löscht die zugehörige Datei. Damit die Daten nicht verloren gehen, werden sie vor dem Löschen durch `protocolwrite/0` in der Datei `protokoll.sve` gesichert.

Mit `style_check/1` werden die gewünschten Fehlermeldungen eingestellt. Danach wird eine Liste von Dateien geladen. Der Aufruf in einer Liste entspricht dem jeweiligen Aufruf der Dateien durch `consult(`$'<$ `Datei` $>'$`)`. Das Prädikat `consult_facts/0` liest nun den Inhalt dieser Dateien in die Datenbasis von Prolog ein. Die Regeln und die möglichen Handlungen werden gezählt und ihre Anzahl wird gespeichert.

Durch Eingabe in der Parameterdatei para.prolog hat man die Möglichkeit die Simulation mit konkret eingegebenen Daten laufen zu lassen. Dazu muß in der Parameterdatei `use_old_data(X)/1` die Variable `(X)` mit `yes` belegt werden. Ist dies nicht der Fall, wird ein zufälliger Charakter erstellt.

Mit `put(x)` erfolgt eine Ausgabe des ASCII–Zeichens `x` auf den aktuellen Ausgabe-

strom[234]. `put(7)` in Zeile 22 erzeugt einen Ton des internen Lautsprechers. Die nächsten Abschnitte[235] sind nötig, um den zufälligen Charakter zu erstellen.

7.2.2 Schleifen

Nach Erstellung der zufälligen Charaktere oder dem Einlesen vorgegebener Charaktere aus der Datei charakter.prolog beginnen die beiden Hauptschleifen des Programms. Wie in SMASS[236] gibt es eine Schleife für `runs` und eine Schleife für Perioden (`periods`). Schleifen unter Prolog sind am einfachsten mit dem Prädikat `between/3` zu programmieren. Die Struktur ist folgendermaßen:

```
(between(N1,N2,X),
preds(...,X,...),
fail
; true
).
```

Eine andere Möglichkeit zur Programmierung von Schleifen besteht mit dem Prädikat `repeat/0`. Dazu ein Beispiel:

```
zaehle_bis :-
  integer(Grenze),
  Grenze >= 0,
  abolish(zaehler,1),
  assert(zaehler(0)),
  repeat,      /*Beginn des Schleifenrumpfes/*
    retract(zaehler(N)),
    NNeu is N+1,
    assert(zaehler(NNeu)),
    write(N),nl,
  N == Grenze,
  !.
```

[234] zum Ein– und Ausgabestrom siehe 5.4.5 auf Seite 59.
[235] siehe Programmzeile 63 bis Zeile 156 ab Seite XCVII im Programm stat.prolog.
[236] siehe Abschnitt 6.3 auf Seite 75.

Diese Konstruktion ist äquivalent zum „Generieren und Testen“, das auf Seite 57 vorgestellt wurde. Ebenso wie dort ist der Cut als Abschluß der Bedingungsanfrage notwendig, da der Schleifenrumpf beliebig oft erfüllbar ist[237]. Durch `repeat` werden Prädikate genauso wiederholt ausgeführt, wie es bei der Rekursion in der Programmabarbeitung geschieht.
Die Schleife des Prädikats `begin/0` ist eine Schleife, die mit `between/3` gebildet wird. Diese Schleife erzeugt die `runs`. Diese äußere Schleife dient einigen statistischen Eigenschaften. Ohne Veränderung der Charaktere wird diese Schleife mehrmals durchlaufen. Die Spezifikation der Anzahl von Wiederholungen erfolgt durch den Benutzer.
Die zweite Schleife muß mit `repeat/0` konstruiert werden. Die innere Schleife gibt den Agenten die Möglichkeit zu handeln. Die maximale Anzahl von Handlungen wird vom Benutzer vorgegeben. Die Schleife kann aus drei Gründen abgebrochen werden:

1) Einer der Agenten gibt auf.
2) Einer der Agenten beginnt einen Großangriff[238].
3) Die maximale Anzahl der Perioden, wie sie vom Benutzer vorgegeben wurde, ist erreicht.

Um offene *Loops* zu vermeiden, kann diese Schleife nicht mit `between/3` aufgebaut werden. Es muß die Konstruktion mit `repeat/0` verwendet werden. In der äußeren Schleife wird die innere Schleife mit `mainloop/3` aufgerufen.

[237] siehe Wilhelm Weisweber, [120], 1997, Seite 84.
[238] Der Ausgang des Großangriffs wird nicht untersucht. Die maximale Eskalationsstufe ist erreicht. Für die Krisensimulation ist der Ausgang des Großangriffs irrelevant.

7.2.3 Entscheidungstabelle

Die innere Schleife benutzt eine Entscheidungstabelle. Prolog bietet den Vorteil, Entscheidungstabellen direkt in Prolog–Klauseln zu übersetzen. Die entsprechende Prolog–Notation wird als geschützte Klausel (*guarded clause*) bezeichnet. Die Bedingungen *bX* werden abgeprüft, um eine Aktion *aX* auszuführen. Um bei mehreren *guarded clause* Regeln eine indeterministische Lösung zu vermeiden, muß zwischen den Bedingungen und Aktionen ein Cut eingefügt werden.

	1	2	3	4	5	6	7	8
b1	j	j	j	j	j	n	n	–
b2	j	n	j	j	n	j	n	–
b3	j	–	n	–	–	–	–	–
b4	n	n	n	j	j	j	–	–
a1	x	–	–	–	x	x	–	–
a2	–	x	x	x	–	x	–	–
a3	x	x	–	x	–	x	x	–
a4	–	–	–	–	–	–	–	x

Tabelle 11: Entscheidungstabelle nach Peter Schnupp, [113], 1989, Seite 142.

Die hier gewählte Lösung ist eine Eintrefferentscheidungstabelle. Das bedeutet für die Abarbeitung, daß eine der Regel zugeordnete Aktion ausgeführt wird, sobald eine Regel erfüllt ist. Anschließend geht das Programm wieder in den übergeordneten *Loop*. Ob noch weitere Regeln erfüllt sind, wird nicht untersucht. Die Anordnung der Regeln ist deshalb von zentraler Bedeutung. Die Tabelle 11 entspricht folgendem Prologprogramm:

```
/*1*/  tabelle1 :- b1,b2,b3,notb4,a1,a3.
/*2*/  tabelle1 :- b1,notb2,notb4,a2,a3.
/*3*/  tabelle1 :- b1,b2,notb3,notb4,a2.
/*4*/  tabelle1 :- b1,b2,b4,a2,a3.
/*5*/  tabelle1 :- b1,notb2,b4,a1.
/*6*/  tabelle1 :- notb1,b2,b4,a1,a2,a3.
/*7*/  tabelle1 :- notb1,notb2,a3.
/*8*/  tabelle1 :- a4.
```

Durch Einfügen von Variablen läßt sich die Entscheidungstabelle erweitern. Statt der Wahrheitswerte *j*, *n* und −− (gleichgültig) werden unmittelbare Vergleiche eingetragen. Die gewünschte Aktion ersetzt das *x*.
In Entscheidungstabellen sind bei den Regeln prinzipiell vier Beziehungstypen möglich:

- Gleichheit (Identität),
- Ausschluß (Exklusion),
- Einschluß (Inklusion) und
- Überschneidung (Intersektion).

Im vorliegenden Programm werden nur wenige Regeln verwendet. Sollte jedoch diese Regelbasis erheblich erweitert werden, stehen komfortable Entscheidungstabellensysteme zur Verfügung, die Verfahren durchführen, um die Tabellen zu verbessern. In diesen Systemen werden dann zum Beispiel redundante Regeln gelöscht[239]
Der Aufruf der Entscheidungstabelle erfolgt mit `tabelle_1/3`. Dieses Prädikat verwendet zur Abarbeitung die maximale Anzahl der Perioden (`TT`), die zukünftige Periode (`T + 1`) und den ausgeführten Run (`R`). Diese Parameter werden dem dreistelligen Prädikat von der Hauptschleife mitgegeben. Falls nötig können auf einfache Weise zusätzliche Informationen hinzugeladen werden.
Wichtige Beispiele für das Laden relevanter Informationen sind:

- Anzahl der Akteure (`selected_actors/2`),
- Charakter der Akteure (`charakter/4`)[240] und
- die vorhergehende Handlung (`vorher_contr/2`).

Die Tabelle kontrolliert zuerst, ob die vorherige Handlung nicht zum Abbruch der Periode führt. Sind die vorherigen Handlungen `aufgeben` oder `grossangriff`, wird die Periode

[239] siehe Gerald Jüttner, [68], 1989, sowie Michael Rammé, [106], 1996.

[240] Bei dieser Konstruktion wird auch das Systemprädikat `once/1` verwendet. Siehe Abschnitt 5.4.4 auf Seite 56.

abgebrochen. Bei Abbruch ist diese Periode beendet. Im zweiten Schritt wird festgestellt, ob die Bedingungen für `aufgeben` erfüllt sind. In dieser Version ist das der Fall, wenn

- der Gegner erheblich stärker ist ($Staerke1 - Staerke2 > 90$) oder
- der Gegner stärker ist und erheblich eskaliert.
  ($Staerke1 - Staerke2 > 70$ und die vorherige Handlung des anderen Akteurs ist zwei Eskalationsstufen höher, als die davorliegende letzte Handlung des jetzigen Akteurs. Da in dieser Simulation die Akteure nicht abwechselnd handeln, kann es durchaus sein, daß der Gegner zweimal eskaliert. Auch in diesem Fall wird der Programmlauf mit `aufgeben` beendet.).

Die Entscheidungstabelle kommt genau folgender Forderung nach:

> „ ... zur Unterstützung des Entscheidungsprozesses in Krisen, wird man sich speziellere Simulationen *S* wünschen, bei denen die Ziele und die gewählten Pläne nicht „konstant“ bleiben, sondern sich mit den wechselnden Umständen auch ändern können.“[241]

Berücksichtigung finden auch die von Sander geforderten Persönlichkeitsfaktoren (Charaktere)

> „ ... unter Berücksichtigung von vorher festgelegten Entscheidungsregeln ... “[242].

Die nächsten beiden Bedingungen prüfen, ob die Periode abgebrochen werden muß, weil die maximale Periodenanzahl überschritten wurde. Als letzte Bedingung steht die *Default-Regel*, die ausgeführt wird, wenn keine der vorherstehenden Regeln zutrifft.
Es ist leicht zu sehen, wie dieses Konzept zu erweitern ist. Das Prädikat `tabelle_1/3`. ruft das Prädikat `kernel/2` auf.

[241] Jörg Sander, [111], 1993, Seite 65.
[242] Jörg Sander, [111], 1993, Seite 65.

7.2.4 Charaktere

Durch `kernel/2` wird eine Handlung abhängig vom Charakter des Akteurs ausgewählt. Wie bei SMASS[243] wird der Charakter eines Akteurs A durch eine Liste $[C_1,\ldots,C_{MM}]$ beschrieben, wobei für jede Regel M eine Zahl C_M den Ausdruck von As Charakter in Bezug auf die Regel M repräsentiert.

Wenn zum Beispiel die Regel für Eskalation M ist, zeigt der Wert von $C_M (0 \leq C_M \leq 100)$ As Neigung an, zu eskalieren. Eine aggressiver Akteur wird den Wert 100 $(C_M = 100)$ haben[244]. Dabei sucht das Prädikat `consult_profile/1` zuerst die Regel, nach welcher der Akteur handeln soll. Die Auswahl der Regel wird wie folgt durchgeführt: Sei $\Theta = \sum_{i<MM} C_i$, wobei $C_i,\ldots,C_{MM}$ Zahlen sind, die in As Charakter $C = [C,\ldots,C_M M]$ vorkommen. Aus der Menge $\{1,\ldots,\Theta\}$ wird eine Zufallszahl, abhängig von der diskreten Verteilung der Gewichte $C_1,\ldots,C_{MM}$, gezogen. Wenn die Zufallszahl z.B. R in der Menge $\{1,\ldots,C_1\}$ liegt, wird die Regel 1 angewendet. Wenn R in die Menge $\{C_1+1,\ldots,C_2\}$ fällt, wird die Regel 2 benutzt und so weiter. Die Werte von As Charakter werden für die Wahl der Regel benutzt.

Wenn es zum Beispiel drei Regeln gibt und A den Charakter $[2,1,4]$ hat, dann wird mit einer Wahrscheinlichkeit von $\frac{2}{7}$ die Regel 1 gewählt und mit einer Wahrscheinlichkeit von $\frac{1}{7}$ Regel 2 und mit einer Wahrscheinlichkeit von $\frac{4}{7}$ die Regel 3.

Die Summe der Charakterverteilung eines Akteurs ergibt 100 ($\Theta = 100$). Somit gilt $100 = \sum_{i\leq R} C_i$, wobei R die Anzahl der benutzten Regeln ist, die im Charakter jedes Akteurs $C = [C_1,\ldots,C_R]$ vorkommen. Abhängig vom Charakter werden so die Handlungen ausgewählt. Nach dem Schreiben der ausgewählten Handlung in die Datenbasis wird der Wert für die Perioden durch `adjust/1` um eins heraufgesetzt. Das Schreiben der Handlungen unterscheidet sich bei der ersten und letzten Handlung vom Normalfall. Dafür existieren folgende Prädikate:

[243] siehe Abschnitt 6.3.2 auf Seite 80.

[244] Die Daten der Gewichte unterliegen einer diskreten Verteilung. Der Charakter ist in jedem Agent diskret verteilt.

- `write_initial/1` für die erste Handlung,
- `write_action/9` für den Normalfall und
- je nach Abbruchbedingung für das Ende der Perioden:
  - `ende_normal/3` für die maximale Anzahl der Perioden und
  - `end/2` in Kombination mit `ende/2` für alle anderen Fälle.

7.2.5 Regeln

Jede Regel befindet sich in einer eigenen Datei. Mit dieser Konstruktion ist es einfach neue Regeln hinzuzufügen. Die Regeln, die das Programm verwenden soll, sind in der Datei rules.prolog verzeichnet. Die Dateien sind:

- linear,
- lingr,
- lin2,
- desk,
- deskgr,
- desk2,
- titfortat,
- tfts und
- maxesk.

Die erste Regel beschreibt die lineare Eskalation über alle Handlungen:

```
1  linear(R,T,AS,Staerke,Nr):-
2    Vorher is T-1,
3    action(R,Vorher,_,Direktion,Handlungsname,Staerke,Maxstaerke,Nr),
4    Staerke<Maxstaerke,
5    NS is Staerke+1,
6    Handlung=Handlungsname,
7    direktion_ini,
8    direktion(D),
9    asserta(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),
10   (retract(choosed_actionart(act(_,1,_,_))); true),
11   asserta(choosed_actionart(act(Handlung,1,Maxstaerke,Nr))),
12   feasable(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),!.
13
15 linear(R,T,AS,Staerke,Nr):-
16   Vorher is T-1,
17   action(R,Vorher,_,Direktion,Handlungsname,Staerke,Maxstaerke,Nr),
```

```
18  Staerke = Maxstaerke,
19  esk(Handlungsname,Handlung),
20  act(Handlung,1,X,Num),
21  direktion_ini,
22  direktion(D),
23  asserta(action(R,T,AS,D,Handlung,1,X,Num)),
24  (retract(choosed_actionart(act(_,1,_,_))); true),
25  asserta(choosed_actionart(act(Handlung,1,Max,Num))),
26  feasable(action(R,T,AS,D,Handlung,1,Max,Num)),!.
```

Bei dieser Regel gibt es eine Fallunterscheidung. Den ersten Fall beschreiben Zeile 1 bis Zeile 12. Der zweite Fall ist von Zeile 15 bis Zeile 26 programmiert. Jede Handlung besitzt einen vom Benutzter eingegebenen maximalen Stärkegrad. Ist der Stärkegrad der zuvor ausgeführten Handlung geringer als der maximale Stärkegrad, wird die Handlungsstärke einfach um eins erhöht. Ist der vorhergehende Stärkegrad gleich dem maximalen Stärkegrad, wird der Handlungstyp gewechselt. Nach der Tabelle 20 auf Seite XXIV im Anhang wird die nächststärkere Handlung ausgewählt.
Die zweite Regel funktioniert genauso. Jedoch werden statt Tabelle 20 auf Seite XXIV die Tabellen 21 und 22 auf den Seiten XXV und XXVI verwendet. Wird in diesen Tabellen die maximale Handlung einer Gruppe erreicht, kann durch diese Regel nicht weiter eskaliert werden. Die Datei `lin2` beschreibt eine Eskalation über zwei Handlungsarten hinweg. Bei dieser Regel existiert keine Fallunterscheidung. Der neue Stärkegrad der Handlung ist 1. Analog zu diesen drei Regeln funktionieren die drei Regeln der Deeskalation. Die Tabellen der Eskalation werden auch hier verwendet. Jedoch wird die aufsteigende Richtung durch eine absteigende ersetzt.
Die zwei Regeln für `titfortat` antworten immer mit der gleichen Handlung. Bei der einen Regel entspricht der Stärkegrad dem der vorhergehenden Handlung. Die andere Regel verwendet einen zufälligen Stärkegrad.
Die letzte Regel ist die maximale Eskalation. Wird diese gewählt, wird immer ein Großangriff durchgeführt. Diese Regel muß sehr vorsichtig verwendet werden, um zu vermei-

den, daß zu viele Perioden mit `grossangriff` beendet werden[245].

7.2.6 Prüfung

Die ausgewählte Handlung wird einer erneuten Prüfung unterzogen. Die Kontrolle, ob die auszuführende Handlung überhaupt möglich ist, geschieht mit Aufruf des Prädikats `feasable/1`, das in der Datei feasable.prolog definiert wird. Stellt sich heraus, daß die gewünschte Handlung nicht möglich ist, muß diese Handlung unterbrochen und eine neue ausgewählt werden. Hier liegt ein Handlungskonflikt vor. Die betreffende Aufgabe kann nicht mehr erfüllt werden.

> „Die „Leiter“ und die „Decke“ können nicht von der gleichen Person zur gleichen Zeit gestrichen werden. Wird die Abfolge der Handlungen „Leiter streichen“ und „Decke streichen“ nicht beachtet, so kann ein Handlungskonflikt entstehen. ... Die Lösung eines potentiellen Handlungskonflikts in einer gegebenen Aufgabe sehen wir als elementare Planleistung an.“[246]

Die Standardeinstellung für das Prädikat `feasable/1` ist `true`. Wird dieses Prädikat genutzt, können unerwünschte Folgehandlungen unterdrückt werden. Soll beispielsweise eine weitere Angriffsdrohung nach einem Angriff vermieden werden, ist die Abweisung einer Angriffsdrohung in diese Datei einzutragen[247].

[245] siehe auch Abschnitt 7.3 auf Seite 112.

[246] Frank H. Piekara, [104], 1985 Seite 4. Das Beispiel stammt von E. D. Sacerdoti, [110], 1977, der das Programm NOHA (Nets Of Actions Hierarchies) entwickelte.

[247] Im aktuellen Programm ist eine Angriffsdrohung nach einem Angriff erlaubt!

7.2.7 Statistik

Die Datei stat.prolog enthält ein Programm, um automatisch verschiedene Charaktere zu testen. In der Datei charakters.prolog werden sieben verschiedene Charaktere festgelegt. Jedem Charakter wird ein Wert zugeordnet, der die Wahrscheinlichkeit angibt, mit der dieser Charakter eine bestimmte Regel auswählt. Um den Inhalt der Zahlenkolonne in der Liste leichter zu verstehen, wird jedem Charakter ein Name mitgegeben. Aus dem Charakternamen läßt sich leicht die Intention der Liste erschließen.

	a	b	c	d
a	~~aa~~	~~ab~~	~~ac~~	~~ad~~
b	ba	~~bb~~	~~bc~~	~~bd~~
c	ca	cb	~~cc~~	~~cd~~
d	da	db	dc	~~dd~~

Abbildung 16: Kombinationen einer vierelementigen Menge.

```
charakter_arts(agressiv,[30,2,2,2,30,2,30,1.95,0.05]).
charakter_arts(ausgeglichen,[12,13,13,13,12,12,12,12.99,0.01]).
charakter_arts(deeskalierend,[1,6,5,30,1,30,1,30.99,0.01]).
charakter_arts(tft,[4,40,40,3,3,3,3,3.99,0.01]).
charakter_arts(tftdesk,[1,19,20,19,1,19,1,19.99,0.01]).
charakter_arts(tftesk,[19,20,20,1,19,1,19,0.99,0.01]).
charakter_arts(absolutdeeskalierend,[0,0,0,35,0,35,0,29.99,0.01]).
```

Der Akteur definiert sich aus dem Charakter und seiner Stärke. Ein weiteres Merkmal für Akteure ist Fanatismus. Fanatische Akteure haben die Handlung `aufgeben` nicht im Handlungsarsenal.
Ein Beispiel für einen Akteur ist folgendes:

```
([30,2,2,2,30,2,30,1.95,0.05] 10 yes).
```

Die Interpretation der Zahlenreihe ist ein aggressiver, fanatischer Akteur mit der Stärke 10. Um möglichst viele Fälle abzudecken, werden die sieben Charakterarten mit fünf Stärkegraden und der Information über Fanatismus kombiniert. Die drei zu kombinierenden Mengen haben folgendes Aussehen:

- Charakterarten:

$$C = \{aggressiv, ausgeglichen, deeskalierend, tft, tftdesk, tftesk, absolutdeeskalierend\},$$

- Stärkegrade:
  $S = \{0, 10, 50, 75, 100\}$ und
- Fanatismus:
  $F = \{yes, no\}$.

Es treten jeweils zwei Akteure gegeneinander an (*binary crisis*). Dazu werden aus allen Charakterarten Paare herausgegriffen. Diese Paare werden mit Paaren aus der Menge der Stärken und Paaren aus der Menge $\{yes, no\}$ verknüpft. Um alle geordneten Paare zu behandeln, wird jeweils das kartesische Produkt der Mengen gebildet ($M \times M = \{(x,y) \mid x \in M \wedge y \in M\}$). Da keiner der beide Akteure ausgezeichnet ist[248], treten dabei Wiederholungen auf, die eliminiert werden müssen. Dafür wird in der Datei pred.prolog das Prädikat `permut/2` definiert. Auf die erste Menge wird diese Definition angewendet. Bei der zweiten und dritten Menge ist eine Fallunterscheidung nötig. Sind die Elemente des ersten Tupels gleich ((x,x)) wird `permut/2` verwendet, unterscheiden sich die Elemente des ersten Tupels, wird das kartesische Produkt benutzt.
Verständlicher werden die Unterscheidungen bei Betrachten der Abbildung 16 auf Seite 105. Die auf der schrägen Linie liegenden Tupel benötigen das Prädikat `permut/2`. Die im Bereich der gepunkteten Linie liegenden Tupel fallen weg.
Die Datei pred.prolog enthält die Prädikate `permut/2` und `kart_prod/3`. Das kartesische Produkt zweier Mengen A und B errechnet Prolog mit folgendem Programm:

[248] Der Akteur, der mit der initialen Handlung beginnt, wird durch Zufall bestimmt.

```
kart_prod([ ],_,[ ]).

kart_prod([X | R1],L,L3) :-
  paare(X,L,L1),
  kart_prod(R1,L,L2),
  append(L1,L2,L3).

paare(_,[ ],[ ]).

paare(X,[Y | R],[[X,Y] | R1]) :-
  paare(X,R,R1).
```

Der Aufruf erfolgt beispielweise mit `kart_prod([a],[b],X).`. Die Kombinationen von je zwei Elementen aus zwei gleichen Mengen erhält man, indem folgendes Prädikat auf sie angewendet wird:

```
permut(X,Z) :-
  kart_prod(X,X,A),
  sortiere(A,B),
  sort(B,Z).

sortiere(Liste,Rliste) :-
  sort_akku(Liste,[ ],Rliste).

sort_akku([ ],Akku,Akku).

sort_akku([[Head | Tail] | Rest],Akku,Rliste) :-
  (Head = Tail,true;
  msort([Head | Tail],Element1)),
  sort_akku(Rest,[Element1 | Akku],Rliste).
```

Ein Beispiel zum Aufruf ist `permut([a,b,c],X)`. Diese Konstruktion wird verwendet, um den Programmablauf mit Hilfe eines Akkumulators zu beschleunigen.
Bei der Verwendung von Listen ist es oft sinnvoll, Akkumulatoren[249] zu verwenden, um Rechenzeit einzusparen. Prolog erstellt jede Liste zweimal. Zuerst wird die Liste bei der

[249] siehe William F. Clocksin, [33], 1994, Seite 62 – 68.

Ausarbeitung gebildet. Erneut muß die Liste bei der Verrechnung mit einer bestehenden Liste erarbeitet werden.

Ein Akkumulator repräsentiert die bisherigen Ergebnisse, die in Form einer Liste vorliegen. Die Erstellung der Liste und die Berechnung werden in einem Schritt durchgeführt. Zu beachten ist jedoch, daß die Liste in umgekehrter Reihenfolge vorliegt. Um diese Umkehrung zu vermeiden, kann eine Differenzstruktur verwendet werden. Sie ist nichts anderes als ein „Loch“ in einer Liste, in das weitere Elemente eingefügt werden.

7.2.8 Ergebnisdateien

Die Simulationsergebnisse werden in die Dateien history.prolog, ende.prolog, tabellegesamt und Ergebnis geschrieben[250]. Ein Beispiel für die Ausgabe ist das Format eines Charakters in der Datei history.prolog:

```
% ----------Akteur1----------
% Regeln:[linear,titfortat,tfts,desk,lingr,deskgr,lin2,desk2,maxesk]
% Staerke:75Fanatiker:yes
charakter(1,[0,0,0,35,0,35,0,29.99,0.01],75,yes).
```

Die Datei maxende.prolog untersucht die Charaktere, bei denen die maximale Anzahl von Perioden erreicht wurde. Die einfachste, aber auch ineffektivste Verwendung von Zählern verwendet einen in der Datenbasis abgespeicherten Fakt[251].

```
asserta(zaehler(0)),
⋮
inkrement(zaehler(X)):-
  var(X),
  retract(zaehler(X0)),
  X is X0+1,
  asserta(zaehler(X)).
```

Wesentlich effizienter ist das Prädikat `flag(+Key,-Old,+New)`.

```
succeeds_n_times(Goal,Times):-
  flag(succeeds_n_times,Old,0),
  Goal,
  flag(succeeds_n_times,N,N+1),
  fail; flag(succeeds_n_times,Times,Old).
```

[250] zu Ausgabedateien siehe Abschnitt 5.4.5 auf Seite 59.
[251] siehe Peter Schnupp, [113], 1989, Seite 145.

Noch effizienter ist eine Hilfsvariable als Zähler. Das Vorgehen mit Hilfsvariable entspricht auch mehr der Idee von Prolog. Dabei wird von einem Anfangswert rückwärts gezählt.

```
goal(Term) :-
  ⋮
  goal(Term, Zaehler).
goal(Term, Zaehler) :-
  Zaehler > 0,
  ⋮
  Zaehler1 is Zaehler - 1,
  goal(Term, Zaehler1).
goal(Term, 0).
```

Um die Datenfülle zu bewältigen, wird in diesem Programm das Prädikat `flag/3` verwendet. Die Zähler werden in einer eigenen Datei flags.prolog verwaltet. Für jeden Handlungstyp wird ein eigener Zähler generiert:

```
action_flags :-
  ⋮
  findall(Nummer, act(_,_,_,Nummer), Anzahl),
  length(Anzahl, Laenge),
  (between(1, Laenge, X),
  nth1(X, Anzahl, Y),
  act(H,_,_,Y),
  %(between(1,3,Z),
  nth1(1, [positiv,negativ,unterlassung], Dir),
  flagname_run(H, Dir, F),
  flag(F,_,0),
  flagname_gesamt(H, Dir, H1),
  flag(H1,_,0),
  fail; true,!).

flagname_run(H, Dir, F) :-
  concat(H,'_',Hn),
```

```
    concat(Hn,Dir,F).

flagname_gesamt(H,Dir,H1):-
    concat(H,'_',Hn),
    concat(Hn,0,H0),
    concat(H0,Dir,H1).
```

Mit dem Prädikat `concat/3` werden die Namen der Bestandteile des Zählers aneinandergebunden. Zu Beginn jedes Laufes werden alle Zähler auf Null gesetzt. Wird eine Handlung ausgeführt, wird der entsprechende Zähler aufgerufen und um Eins erhöht. Ähnlich wird mit den Zählern für die Abbrüche der Läufe und die Gesamtzahl der `runs` und `periods` verfahren.

7.3 Ergebnisse

Das wichtigste Resultat der Simulation ist die Erkenntnis, daß eine Simulation im Rahmen abstrakter Handlungstypen möglich ist.

Das Grundprogramm terminiert die Hauptschleifen im Mittel nach hundertzwanzig Runs. Wie die Schleifen enden, läßt sich mit den verschiedenen Parametern beeinflussen. Die Parameter (`aufgeben` und `maxesk`) sind die wichtigsten Möglichkeiten zur Einflußnahme. Ein Beispiel für einen Durchlaufen der sieben Charaktere mit 20 Runs zu je 1000 Perioden ist das Ausführen von 1.168.270 Handlungen. Die Anzahl der Runs beträgt 35.700, wobei 18.813 mit „aufgeben“ enden und 16.887 Großangriffe die Handlungsketten beenden. Keinmal wird die maximale Anzahl von Perioden erreicht. 1.785 verschiedene Akteure werden in jedem Lauf miteinander verglichen (Anzahl der Runs/Runs pro Charakter: 35.700/20). Beispielhafte Verteilungen, wie Perioden enden, zeigen die Abbildungen 18 bis 21 auf Seite CVIII.

Die Regel der Maximaleskalation hat natürlicherweise den größten Einfluß auf den Ausgang der Simulation. Ihre Gewichtung in einem Charakter sollte den Wert $\frac{10}{Anzahl\ \ der\ \ Perioden}$ nicht überschreiten, um noch interessante Läufe zu erhalten. Wird der Wert höher angesetzt, ist in fast jedem Lauf mit einer maximalen Eskalation zu rechnen. Genauso kann die Anzahl der Runden, die mit `aufgeben` enden, beeinflußt werden. Wie oben dargestellt wurde, haben die unterschiedlichen Stärkegrade Einfluß auf den Handlungstyp `aufgeben`. Wird der Unterschied der Stärkegrade klein gewählt (bei niedriger Maximaleskalationsstufe), endet jeder Lauf, in dem der schwächere kein Fanatiker ist, mit `aufgeben`.

Sind alle Parameter moderat eingestellt und die Anzahl der Perioden wird auf 1000 erhöht, gibt es immer noch Läufe, welche die maximale Anzahl der Perioden erreichen. Diese hohe Anzahl von Perioden setzt voraus, daß beide beteiligten Akteure Fanatiker sind. Keinem ist es möglich aufzugeben. Desweiteren muß der Run mit einer niedrigen Eskalationstufe beginnen oder die Regelauswahl erfolgt mit mehreren deeskalierenden Regeln in Folge. Diese deeskalierenden Regelfolgen treten meist sehr früh auf. In den

ersten hundert Perioden ist die Handlung `miskredit` meist erreicht. Die Handlungen pendeln sich dann bei dieser Handlung ein, die als niedrigste in der Regelbasis eingetragen ist. In einem Lauf waren von 1000 Handlungen 923 `miskredit`. Trotzdem konnte die Handlung `erschweren` nochmals bei Periode 999 erreicht werden. Es konnten auch Programmläufe beobachtet werden, die nach 375 Perioden mit maximaler Eskalation endet. Von diesen 375 Handlungen waren 210 `miskredit` . Die Interpretation solcher Fälle bedeutet:

- Der Ausgang eines Runs ist nie sicher. Jeder Lauf kann plötzlich eskalieren.

Diese Beobachtung deckt sich mit realen Daten. Am interessantesten ist das Verhältnis der Charakterzüge zur Anzahl der aufgetretenen Handlungstypen (τ). Jede Charakterart besitzt soviel Charakterzüge wie Regeln vorhanden sind ($Charakter(c_1, c_2, \ldots, c_n)$, bei n Regeln, $n \in \mathbb{N}$). Nun werden die Charaktereigenschaften der beiden Akteure A und B untersucht ($Charakter_A(c_1^A, c_2^A, \ldots, c_n^A)$, $Charakter_B(c_1^B, c_2^B, \ldots, c_n^B)$). Der Wert

$$\mu(\tau / c_i^A c_j^B) = \frac{\text{Anzahl der Handlungen vom Typ } \tau}{\text{Anzahl des Auftretens von } c_i^A c_j^B}$$

läßt einen Vergleich zwischen den einzelnen Charaktereigenschaften zu. Ein Ausschnitt der Datei protokoll.sve auf Seite CVII zeigt beispielhaft wenige der 1.785 Akteure, die aufeinander losgelassen werden. Die Abbildung 29 auf Seite CXVI zeichnet eine Kurve in der das Vorkommen der Handlung `aufruf` dargestellt wird. Die Y–Achse zeigt die proportionale Häufigkeit des Vorkommens der Handlung in allen zwanzig Runs, bezogen die Anzahl der jeweiligen Perioden. Die Zahl der X–Achse bezeichnet die Kombination der Akteure. Bei den Akteuren von Zeile 9 bis Zeile 20 in der Datei protokoll.sve tritt die Handlung `aufruf` kaum auf. Bei der siebten Kombination von Akteuren ist ein erstes häufiges Auftreten dieser Handlung zu verzeichnen. Die beiden beteiligten Akteure haben folgendes Aussehen:

```
[0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 no
              [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 50 no
```

Bei Akteurkombination 28 ist das nächste gehäufte Auftreten zu beobachten.
Eine Sonderstellung nehmen die Handlung `aufgeben` und `grossangriff` ein. Falls diese Handlungen ausgewählt wurden, endet die momentane Periode. In den Abbildungen 27 und 31 auf den Seiten CXIV und CXVIII ist ihr Verlauf zu sehen. Beide Handlungen haben bei den ersten Runs ihr Maximum. Je höher die Nummer der Charakterkombination, umso niedriger wird der Wert der Y–Achse. Wenn man die Liste der benutzten Charaktere (siehe Datei charakters.prolog auf Seite LXXX) betrachtet wird auch klar, warum der Verlauf der Kurven so ist. Die ersten Charaktere sind die aggressivsten. Zu Beginn bekriegen sich zwei aggressive Akteure. Die Gesamtzahl der Handlungen ist gering. Beide Akteure dringen auf eine schnelle Entscheidung. Ist einer der Akteure kein Fanatiker wird er aufgeben. In den anderen Fällen findet ein Großangriff statt. Verwendet man die Charaktere der Datei glcha.prolog auf Seite LXXXI ist der Verlauf viel gleichmäßiger (siehe Abbildung 28 auf Seite CXV).
Die anderen Handlungen kann man grob in zwei Gruppen einteilen.

- Die erste Gruppe bestimmt sich wie folgt: Die Handlungen treten nur bei den aggressiven Akteuren gehäuft auf. Alle bei allen anderen Akteuren ist diese Handlung die Ausnahme. Beispiele für solche Handlungen sind `manoever`, `streik` und `zurueckhalten` (Abbildungen auf Seite CXIX, CXXI und CXXII).
- Die zweite Gruppe läßt sich eindeutig bestimmten Charakteren zuordnen. So läßt sich die Handlung `anlagenbau` den Charakterkombinationen 197 bis 308 und die Handlung `mobilmachung` den Charakterkombinationen 96 bis 120 und 344 bis 438 zuordnen (siehe Abbildungen auf Seite CXIII und Seite CXX). Solche Zuordnungen zwischen Charakterbestandteilen und Handlungen lassen sich noch schärfer eingrenzen wenn für die Charaktere die Datei 92cha.prolog zugrundegelegt wird. Ein Beispiel ist die Abbildung 30 auf Seite CXVII.

7.4 Erweiterungen und Alternativen

Der Ansatz von KRIS erscheint erfolgversprechend. Weitere Untersuchungen sind auf jeden Fall lohnenswert. Viele Erweiterungsmöglichkeiten sind schon im Programm eingebaut oder einfach zu implementieren.
Einfache Beispiele dafür sind:

- Die Handlungen werden mit ausgeführten Handlungen, unterlassenen Handlungen und reiner Nichtausführung von Handlungen[252] erweitert. Eine Handlung besteht aus der Handlungsart[253] sowie ihrer Ausrichtung. Die Handlung kann aktiv, reaktiv oder beides sein. Ein Beispiel ist:

  ```
  handlung(X,Y,G):-
    handl(X,Y),
    ausrichtung(G).
  ```

  In diesem Programm sind die Namen für Ausführung der Handlung `positiv`, für Unterlassung `unterlassung` und für bewußte Nichtausführung `negativ`.

- Die Charaktere können beliebig erweitert werden. Durch das Einführen neuer Regeln können den Charakteren weitere Bestandteile zugeordnet werden. Zu beachten ist dabei nur, daß die Summe der Charaktereigenschaften $charakter(c_1, c_2, \ldots, c_n)$ Hundert ergibt ($c_1 + c_2 + \ldots + c_n = 100$). Die Regeln selbst können beliebig genau aufgebaut sein. Jeder Handlung kann eine Folgehandlung zugeordnet werden. Auch Handlungsketten kann eine Folgehandlung zugeordnet werden. Dabei können die Handlungketten auch nach Akteuren unterschieden werden.
  Beispiele für Regelerweiterungen sind:

[252] siehe Abschnitt 4.4 auf Seite 43 zur Nichtausführung von Handlungen.
[253] Siehe zu einem ähnlichen Aufbau von Nominalphrasen mit Artikel und Nomen Urs Egli, [41], 1992, Seite 2.

  - Die Auswahl der Handlung erfolgt immer konstant. Diese Regel wählt wie die Regel der Maximaleskalation eine bestimmte Handlung aus.
  - Die Auswahl der Handlung erfolgt immer zufällig.
  - Bei Initialisierung wird die Regel Deeskalation gewählt. Später ist die gewählte Handlung abhängig von der Handlung des anderen Akteurs. Eskaliert der andere Akteur, wird ebenfalls eskaliert. Greift der andere Akteur zu einer schwächeren Handlung, wird ebenso eine schwächere Handlung ausgewählt.
  - Der Akteur verwendet nur die Eskalation, falls der Gegner zweimal Eskalation gespielt hat. Dieser Akteur ist weniger reizbar, als derjenige, der *titfortat* spielt.
  - Es wird immer deeskaliert, bis der Gegner einmal eskaliert. Ab diesen Zeitpunkt wird nur noch die Eskalation gewählt (ewige Verdammnis).

- Die Anzahl der Akteure ist beliebig erweiterbar.

- Es werden Mailboxen für die Akteure eingerichtet. Schickt der Akteur einen Handlungsauftrag an sich selbst, kann diese Mail als Plan interpretiert werden.

- Die allen Akteuren gemeinsame Datenbank der Geschichte wird aufgelöst. Jeder Akteur hat eine eigene Datenbank. Diese Daten können auch unzureichend sein. Gewisse Handlungen werden vergessen oder falsch wahrgenommen.

Einige der in KRIS gewählten Verwirklichungen dieser Simulation sind nicht zwingend. Für viele Bereiche sind andere Lösungsmöglichkeiten denkbar. Die Entwicklung im Bereich der Computertechnologie geht rasant weiter. Software und Hardware wird ständig verbessert. Deshalb ist damit zu rechnen, daß sich in Zukunft auch andere Lösungsmöglichkeiten bieten. Jedoch ist der Aufbau von KRIS so vielversprechend, daß er weiter verfolgt und neuen Entwicklungen der Technologie angepaßt werden sollt.

7.4.1 Handlungslisten

In einem noch unveröffentlichten Artikel beschreibt Frank Dignum eine Simulation mit Handlungen, in der die Handlungen in Listen aneinandergereiht werden. Jede dieser Listen beschreibt dann einen möglichen Weltverlauf. Diese Beschreibung von Welten erscheint zwar elegant, ist aber mit der heutigen Hardware noch nicht durchführbar. Dignum konnte nur wenige Handlungen beispielhaft simulieren. Aber schon die in KRIS verwendeten 32 Handlungen sind mit diesem Konzept nicht mehr simulierbar.

7.4.2 Ordnungen

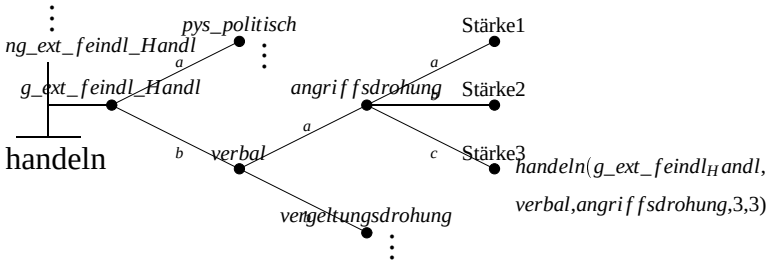

Abbildung 17: Ausschnitt aus dem Handlungsbaum für *handeln*(*g_ext_feindl_Handl*, *verbal*, *angriffsdrohung*, 3, 3).

Die Handlungen können auch mehrstellig aufgebaut werden. Die zusätzlichen Stellen enthalten die Informationen über die Gruppen der jeweiligen Handlungen. Mit einem solchen Aufbau kann die Eskalation der Handlungstypen leichter auf die Gruppen bezogen werden. Ein Beispiel für einen solchen Aufbau zeigt die Tabelle 23 im Anhang auf Seite XXVII. Jedoch ist dieser Ansatz programmtechnisch eingeschränkter, als der in KRIS verwendete. Änderungen in Verbgruppen können nur umständlich programmiert werden[254].

[254] Zu Änderungen in Verbgruppen siehe Abschnitt 3.1 auf Seite 25 und Abschnitt 7.1.1 auf Seite 85.

7.5 Zukunft

Die Idee von KRIS ist so vielversprechend, daß die vorgeschlagenen Erweiterungen unbedingt vorgenommen werden sollte. In wenigen Jahren werden auch die noch bestehenden Beschränkungen durch die Hardware aufgehoben sein, sodaß umfangreiche Krisensimulation durchgeführt werden können. Sicher werden realitätsnahe Krisensimulation in naher Zukunft ausgeschlossen sein. Aber das gezeigte abstrakte Szenario kann Erkenntnisse liefern die in die Krisenforschung einfließen werden.

Anhang

Literaturverzeichnis

Die folgenden Arbeiten (nicht alle werden zitiert oder in den Fußnoten dieser Arbeit erwähnt) sind alphabetisch nach Autoren und Datum der Drucklegung sortiert. Die Arbeiten sind Schriftstücke und Online–Dokumente. Das Akronym URL bedeutet **U***niform* **R***esource* **L***ocator* eine Spezifikation, die standardmäßig im *World Wide Web*[255] für im Internet verfügbare Ressourcen genutzt wird. Wenn nicht anders vermerkt, verweist eine URL nur auf eine Online–Ressource. Eine gedruckte Version ist möglicherweise nicht erhältlich.
Eine ausführlichere Bibliographie zu folgenden Themen findet sich auf meiner Webpage[256]:

- Handlungen,
- soziale Handlungen,
- Krisenforschung,
- Halbordnung,
- Prolog,
- AI und
- soziale Simulationen.

[1] Robert Axelrod. Building new political actors. In: Robert Axelrod, Herausgeber, *The Complexity of Cooperation*, Seiten 120 – 144. Princeton University Press, Princeton, 1997.

[2] Robert Axelrod. *Die Evolution der Kooperation*. Oldenburg Verlag, München, vierte Auflage, 1997.

[255] siehe Rainer Klute, [70], 1996 und Thomas Boutell, [19], 1999.
[256] siehe Dieter Will, [122], 2000.

[3] Tapan P. Bagchi und Vinay K. Chaudhri. *Interactive Relational Database Design*. Springer Verlag, Berlin, 1989.

[4] Thomas T. Ballmer und Waltraud Brennenstuhl. *Deutsche Verben*. Nummer 19 in: Ergebnisse und Methoden moderner Sprachwissenschaft. Gunter Narr Verlag, Tübingen, 1986.

[5] Wolfgang Balzer. A basic model of social institutions. *Journal of Mathematical Sociology*, 16:1 – 29, 1990.

[6] Wolfgang Balzer. *Soziale Institutionen*. Nummer 4 in: Philosphie und Wissenschaft – Transdisziplinäre Studien. de Gruyter, Berlin, 1993.

[7] Wolfgang Balzer. Smass: A serial multi–agent system for simulation in social science. University of Munich, Institut for Philosophy, Logics and Philosophy of Science, March 1996.

[8] Wolfgang Balzer. A theory of binary crises. In: Reiner K. Huber und Rudolf Avenhaus, Herausgeber, *Models for Security Policy in the Post-Cold War Era*, Seiten 233 – 252, Baden–Baden, 1996. Nomos Verlagsgesellschaft.

[9] Wolfgang Balzer. *Die Wissenschaft und ihre Methoden*. Verlag Karl Alber GmbH, München, 1997.

[10] Wolfgang Balzer und Karl Brendel. Dmass: A distributed multi–agent system for simulation in social science. University of Munich, Institut for Philosophy, Logics and Philosophy of Science, April 1996.

[11] Christina Bartl. Künstliche Seelen erkunden ihre Computerwelt – mit Gefühl! URL=http://141.13.70.49/PsiGefuehl.htm, 2000.

[12] Andrew Baum. *Mindstorms*. McGraw Hill, April 2000.

[13] Dave Baum. *Dave Baum's Definitive Guide to* LEGO *Mindstorms*. Apress L.P., 1999.

[14] Henk A. Becker. The emergence of simulation and gaming. *Simulation & Games*, 11(1):11 – 25, 1980.

[15] Jacob Bercovitch. Understanding mediation's role in preventative diplomacy. *Negotiation Journal*, 12, July 1996.

[16] Jacob Bercovitch. International conflict management, 1945 – 1992. URL=http://www.ai.mit.edu/projects/iiip/ir/confman.html, 1999.

[17] Jacob Bercovitch und Richard Jackson. *International Conflict, A Chronological Encyclopedia of Conflicts and Their Management 1945 – 1995*. Congressional Quarterly Inc., Washington D.C., 1997.

[18] Manfred Bierwisch. Semantische und konzeptionelle Repräsentationen lexikalischer Einheiten. *Untersuchungen zur Semantik*, 22:61 – 89, 1983.

[19] Thomas Boutell. World wide web frequently asked questions. Webpage, URL=http://metalab.unc.edu/boutell/faq/wwwfaq.txt, 1999.

[20] Ronald J. Brachman und James G. Schmolze. An overview of the KL–ONE knowledge representation system. *Cognitive Science*, IX:171 – 216, 1985.

[21] Ivan Bratko. *Prolog programming for artificial intelligence*. Addison–Wesley Publishing Company, Singapore, 2. Auflage, 1994.

[22] Michael Brecher. Toward a theorie of international crisis behavior: A preliminary report. *International Studies Quarterly*, 21(1):39 – 74, March 1977.

[23] Michael Brecher. A theoretical approach to international crisis behavior. *Studies in Crisis Behavior*, Seiten 5 – 24, 1979.

[24] Michael Brecher. *Crisis in World Politics, Theory and Reality*. Pergamon Press, Exeter, 1993.

[25] Michael Brecher und Jonathan Wilkenfeld. *Handbook of International Crisis*, Band I von *Crises in the twentieth century*. Pergamon Press, Oxford, erste Auflage, 1988.

[26] Michael Brecher und Jonathan Wilkenfeld. *Handbook of International Crisis*, Band II von *Crises in the twentieth century*. Pergamon Press, Oxford, erste Auflage, 1988.

[27] John M. Brett. The modelling of underwater damage processes for submarin vulnerability studies. In: Michael J. Chinni, Herausgeber, *Military, Government and Aerospace Simulation*, Seiten 68 – 73, April 1995.

[28] A. M. Cameron, C. R. Hoog und C. J. Harris. Flight simulation: Articulated gear leg verses generic gear leg modell. In: Michael J. Chinni, Herausgeber, *Military, Government and Aerospace Simulation*, Seiten 9 – 15, April 1995.

[29] Man Kit Chang und Carson C. Woo. Sanp: A communication level protocol for negotiations. In: Eric Werner und Yves Demazeau, Herausgeber, *Decentralized A.I.*, Band 3, Seiten 31–53. Elsevier Science Publishers B.V., 1992.

[30] Roderick M. Chisholm. *Person and Object. A Metapysical Study*. Allen & Unwin, London, 1976.

[31] Alonzo Church. *Introduction to mathematical logic*, Band 1 von *Princeton mathematical series 17*. Princeton University Press, Princeton/New Jersey, 1956.

[32] Paul M. Churchland. Der logische Status von Handlungserklärungen. In: Ansgar Beckermann, Herausgeber, *Analytische Handlungstheorie*, Band 2 von *Suhrkamp Taschenbuch Wissenschaft 489*, Seiten 304 –331. Suhrkamp Verlag, Frankfurt am Main, 1977.

[33] William F. Clocksin und Christopher S. Mellish. *Programming in Prolog*. Springer Verlag, Berlin, Heidelberg, New York, vierte Auflage, 1994.

[34] Rosaria Conte und Nigel Gilbert, Herausgeber. *Artificial societies, The computer simulation of social life*. UCL Press, London, 1995.

[35] Ralf Cordes, Rudolf Kruse, Horst Langendörfer und Heinrich Rust. *Prolog, Eine methodische Einführung*. Vieweg Verlag, Braunschweig, Wiesbaden, 1992.

[36] J. Barton Cunningham. Assumption underlaying the use of different types of simulations. *Simulation & Games*, 15(2):213 – 234, 1984.

[37] Kathleen Dalgren. *Naive Semantics for Natural Language Processing*. Kluwer Academic Publishers, Boston, 1988.

[38] Matthias Kalle Dalheimer. *Linux, Wegweiser zur Programmierung & Entwicklung*. O'Reilly Verlag, Köln, 1997.

[39] Dietrich Dörner. EMOREGUL. Memorandum 2, Lehrstuhl Psychologie II, Universität Bamberg, Bamberg, April 1994.

[40] Dietrich Dörner. Die PSI–Homepage. URL=http://www.uni.bamberg.de/∼ba2dp1/psi.htm, 1999.

[41] Urs Egli und Christoph Schwarze. Grammatikschreiben in Prolog – Dokumentation zu einem Konstanzer Prolog–Kurs. Arbeitspapier, Fachgruppe Sprachwissenschaft der Universität Konstanz, Konstanz, 1992.

[42] M. van Emden. Red and green cuts. *Logic Programming Newsletter*, 2, 1982.

[43] Andreas Engel und Michael Möhring. Der Beitrag der sozialwissenschaftlichen Informatik zur sozialwissenschaftlichen Modellbildung und Simulation. In: Matthias Gsänger und Jörg Klawitter, Herausgeber, *Modellbildung und Simulation in den Sozialwissenschaften*, Band 13 von *Forum für interdisziplinäre Forschung*, Seiten 39 – 59. Dr. Josef H. Röll Verlag, Dettelbach, 1995.

[44] Armin Ertl. *Prolog, verstehen und anwenden*. IWT Verlag GmbH, Vaterstetten, 1988.

[45] Manfred Flury. Krisen und Konflikte – Grundlagen. Grundlagenforschung 20, Geographisches Institut der Universität Bern, Geographica Bernensia, 1983.

[46] Jay Wright Forrester. *Principles of system*. MIT Press, Cambridge, 1980.

[47] Wilhelm Fucks. *Formeln zur Macht*. rororo, Hamburg, 1967.

[48] Johannes Fürnkranz, Johann Petrak, Robert Trappl und Jacob Bercovitch. Machine learning methods for international conflict database: A case study in predicting mediation outcome. Webpage TR–94–33, Austrian Research Institute for Artificial Intelligence, Wien, 1994. URL=http://www.ai.univie.ac.at/cgi-bin/tr-online?number+94-33.

[49] Iván Futó und Tamaás Gergely. *Artificial Intelligence in Simulation*. Ellis Horwood series in artificial intelligence. Ellis Horwood Limited, New York, 1990.

[50] Johan Galtung. Peace thinking. In: A. Lepawski, Herausgeber, *The Search for World Order*. Appelton–Century–Crofts, New York, 1971.

[51] Klaus Jürgen Gantzel, Torsten Schwinghammer und Jens Siegelberg. Kriege der Welt. *Interdepenz*, 13, 1992.

[52] Hartmann J. Genrich. Ein systemtheoretischer Beitrag zur Handlungslogik. In: Hans Lenk, Herausgeber, *Handlungstheorien – interdisziplinär, Handlungslogik, formale und sprachwissenschaftliche Handlungstheorien*, Band 1 von *Kritische Information, Nr. 62*, Seiten 107 – 136. Wilhelm Fink Verlag, München, 1980.

[53] Michael Georgeff. A theory of action for multiagent planning. In: *Proceedings of the Fourth National Conference on Artificial Intelligence*, Seiten 121 – 125, 1984.

[54] Clifford German. A tentative evaluation of world power. *Conflict Resolution*, 4(1):138 – 144, 1960.

[55] Ulrich Geske. *Prolog*. Akademie Verlag GmbH, Berlin, 1993.

[56] Nigel Gilbert. Multi-level simulation in Lisp–Stat. *Journal of Artificial Societies and Social Simulation*, 2(1):1–7, 1999.
URL=http://www.soc.surrey.ac.uk/Jasss/2/1/3.html.

[57] Carl Ginet. *On action*. Cambridge University Press, Cambridge, 1990.

[58] Alvin Goldman. *A Theory of Human Action*. Engelwood Cliffs, N.J., 1970.

[59] Alvin Goldman. The individuation of action. *The journal of Philosophy*, 68:761 – 774, 1971.

[60] St. Greenwood, 1984.

[61] Bob Guerra. *The Electronic Battlefield*. Compute! Publications, Greensboro, 1987.

[62] Richard Hauser, Uwe Hochmuth und Johannes Schwargu. *Mikroanalytische Grundlagen der Gesellschaftspolitik*. Akademie Verlag, Berlin, 1993.

[63] Rainer Hegselmann. Cellular automata in the social science. In: Rainer Hegselmann, Herausgeber, *Modelling and Simulation in the Social Science from the Philosophy of Science Point of View*, Seiten 209 – 233. Kluwer Academic Publishers, Dordrecht, 1996.

[64] Charles F. Hermann. Some issues in the study of international crises. In: Charles F. Hermann, Herausgeber, *International Crises. Insigth from Behavioral Research*, Seiten 3 – 17. Free Press, New York, 1972.

[65] Hans Hermes. *Einführung in die Verbandstheorie*. Springer Verlag, Berlin, Heidelberg, 1967.

[66] Valerie M. Hudson, Herausgeber. *Artificial Intelligence and International Politics*. Westview Press, Boulder, 1991.

[67] Richard C. Jeffrey. *Logik der Entscheidung*. R. Oldenburg Verlag, Wien, 1965.

[68] Gerald Jüttner und Hardy Feller. *Entscheidungstabellen und wissensbasierte Systeme*. R. Oldenburg Verlag, Wien, 1989.

[69] Herman Kahn. *On Escalation*. Praeger, New York, 1965.

[70] Rainer Klute. *Das World Wide Web: Web–Server und –Clients*. Addison–Wesley Publishing Company, Bonn, 1996.

[71] Jonathan B. Knudsen. *The unofficial Guide to* LEGO *Mindstorms*. O'Reilly & Associates Inc., Sebastopol, erste Auflage, 1999.

[72] Robert A. Kowalski. Algorithm = logic + contol. *ACM*, 22(7):424 – 436, July 1979.

[73] Robert A. Kowalski. *Logic for Problem Solving*, Band 7 von *Artificial intelligence series*. Elsevier Science Publishing, New York, 1979.

[74] Heinz Krummenacher. *Internationale Normen und Krisen. Die normative Dimension internationaler Politik*, Band 11 von *Züricher Beiträge zur Politischen Wissenschaft*. Verlag Rüegger, Grüsch, 1984.

[75] Robert H. Kupperman, Andrew C. Goldberg, Debra van Opstal, Michael E. Brown und James H. Barkley. Leader and crisis: The CSIS crisis simulations. Technischer Bericht 9/5, The center for Strategic and International Studies, Washington D. C., 1987. Significant Issues Series.

[76] Klaus-Jürgen Langer. Entwurf und Implementierung eines Simulationssystems mit integrierter Modellbank– und Experimentdatenverwaltung. Arbeitsbericht 17, Institut für Mathematische Maschinen und Datenverarbeitung, Erlangen, Dezember 1989. Band 22.

[77] Richard Ned Lebow. *Between Peace and War. The Nature of International Crisis*. John Hopkins University Press, Baltimore, London, 1981.

[78] Hans Lenk. Graphen– und Verbandsstrukturen in formalen Handlungstheorien. In: Hans Lenk, Herausgeber, *Handlungstheorien – interdisziplinär, Handlungslogik, formale und sprachwissenschaftliche Handlungstheorien*, Band 1 von *Kritische Information, Nr. 62*, Seiten 137 – 166. Wilhelm Fink Verlag, München, 1980.

[79] Wim Liebrand und David Messick. Computer simulations of sustainable cooperation in social dilemmas. In: Rainer Hegselmann, Herausgeber, *Modelling and Simulation in the Social Science from the Philosophy of Science Point of View*, Seiten 235 – 247. Kluwer Academic Publishers, Dordrecht, 1996.

[80] R. W. Mack und R. C. Snyder. The analysis of social conflict: Towards an overview and synthesis. *Journal of Conflict Resolution*, 1(2):212 – 248, 1957.

[81] Erich Mater. *Deutsche Verben*, Band 1. VEB Bibliographisches Institut, Leipzig, 1966.

[82] Bruce Mayo. Describing verbs of motion in prolog. Arbeitspapier, Forschergruppe „Das Lexikon in der Organisation der Sprache“, Konstanz, Oktober 1991.

[83] John McCarthy. *Defending AI research*, Band 49 von *CLSI lecture notes*. Center for the Study of Language and Information, Stanford, 1996.

[84] Charles A. McClelland. Access to Berlin: The quantity and variety of events, 1948 – 1963. In: *Quantitative International Politics: Insights and Evidence*, Seiten 159 – 186. The Free Press, New York, 1968.

[85] Dennis Meadows. *Die Grenzen des Wachstums*. Deutsche Verlags Anstalt, Stuttgart, 1972.

[86] Dwain Mefford. Steps toward artificial intelligence: Rule–based, case–based, and explanation–based models of politics. In: Valerie M. Hudson, Herausgeber, *Artificial Intelligence and International Politics*, Seiten 56 – 96. Westview Press, Boulder, 1991.

[87] Jürgen Mittelstraß. *Enzyklopädie Philosophie und Wissenschaftstheorie*. J. B. Metzler Verlag, Stuttgart, 1995.

[88] Michael Möhring. MIMOSE, *Eine funktionale Sprache zur Beschreibung und Simulation individuellen Verhaltens in interagierenden Populationen*. Dissertation, Institut für Sozialwissenschaftliche Informatik, Koblenz–Landau, 1990.

[89] Joseph J. Molitoris. Battle managers, military simulation, and virtual reality. In: Michael J. Chinni, Herausgeber, *Military, Government and Aerospace Simulation*, Seiten 135 – 141, April 1995.

[90] Beat Moser. Bürgerkrieg und bürgerkriegsähnliche Auseinandersetzungen. Kleine Studien zur Politischen Wissenschaft 30, Forschungsstelle für Politische Wissenschaft, Universität Zürich, 1974.

[91] Jörg P. Müller. The right agent (architecture) to do the right thing. In: Jörg P. Müller, Herausgeber, *5th International Workshop, Paris*, Band V von *Intelligent Agents*, Seiten 211 – 226, Berlin, Juli 1999. Springer Verlag.

[92] Andrej Nowak und Maciej Lewenstein. Modelling social changes with cellular automata. In: Rainer Hegselmann, Herausgeber, *Modelling and Simulation in the*

Social Science from the Philosophy of Science Point of View, Seiten 249 – 285. Kluwer Academic Publishers, Dordrecht, 1996.

[93] Office of Technology Assessment, Congress of the United States. *Virtual reality and technologies for combat simulation*. Background paper. US Government Print Office, Washington, DC, 1994.

[94] Peter Pause, Achim Botz und Markus Egg. Lexical variation and representation. Arbeitspapier, Forschergruppe „Das Lexikon in der Organisation der Sprache“, Konstanz, 1991.

[95] John Perry. The problem of the essential indexical. *Noûs*, 13:3 – 21, 1979.

[96] Johann Petrak, Robert Trappl und Johannes Fürnkranz. The possible constribution of AI to the avoidance of crisis and war: Using CBR methods with the KOSIMO data base of conflicts. Webpage TR–94–32, Austrian Research Institute for Artificial Intelligence, Wien, 1994.
URL=http://www.ai.univie.ac.at/cgi-bin/tr-online?number+94-32.

[97] Karl Pfeifer. *Actions and Other Events. The Unifer–Multiplier Controvery*. Lang, New York, 1989.

[98] Frank Pfetsch. Krieg und Frieden in neuerer Zeit. In: Frank Pfetsch, Lothar Burchardt, Günther Roth und Detlev Junker, Herausgeber, *Wie Kriege entstehen*, Band 8 von *Schriftenreihe des Volksbundes Deutsche Kriegsgräberfürsorge*, Stuttgart, 1990. Landeszentrale für politische Bildung Baden–Würtemberg.

[99] Frank Pfetsch, Herausgeber. *Konflikte seit 1945*. Ploetz Verlag, Freiburg, 1991.

[100] Frank Pfetsch, Herausgeber. *Handlung und Reflexion: theoretische Dimensionen des Politischen*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1995.

[101] Frank Pfetsch, Herausgeber. *Globales Konfliktpanorama 1990 – 1995*. Lit Verlag, Münster, 1996.

[102] Frank Pfetsch. *KOSIMO*. Heidelberg, 1999.
URL=http://www.conflict.com/hiik/Home_German.htm.

[103] Frank Pfetsch und Peter Billing. *Datenhandbuch nationaler und internationaler Konflikte*. Nomos Verlagsgesellschaft, Baden–Baden, 1994.

[104] Frank H. Piekara. Bewältigung eines Handlungskonfliktes in einem Modell zur Wissensrepräsentation. GMD – Studien 65, Gesellschaft für Mathematik und Datenverarbeitung MBH Bonn, Schloss Birlinghoven, St. Augustin, 1982.

[105] A. Pugh. DYNAMO *User's Manual*. MIT Press, Cambridge, 1976.

[106] Michael Rammé. *Entscheidungstabellen – Entscheiden mit System*. Prentice Hall Verlag GmbH, München, 1996.

[107] Anand S. Rao und Michael P. Georgeff. Modelling rational agents within a BDI architecture. In: James Allen, Herausgeber, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Seiten 473 – 484, San Mateo, California, 1991. Morgan Kaufmann Publishers Inc.

[108] J. Lewis Rasmussen und Robert B. Oakley. *Conflict Resolution in the Middle East: Simulating a Diplomatic Negotiation Between Israel and Syria*. United States Institute of Peace Press, Washington D.C., 1992.

[109] Edmund Runggaldier. *Was sind Handlungen?*, Band 12 von *Münchner philosophische Studien – Neue Folge*. Kohlhammer Verlag, Stuttgart, 1996.

[110] Earl D. Sacerdoti. *A structure for plans and behavior*. Elsevier, New York, 1977.

[111] Jörg Sander. Theoretische Grundlagen für rechnergestützte Krisensimulation und –Beratung. Forschungsberichte 44, Institut für Statistik und Wissenschaftstheorie, München, 1993.

[112] Peter Schnupp. *Prolog*. Carl Hanser Verlag, München, 1986.

[113] Peter Schnupp. *TerminalBuch Prolog*. R. Oldenburg Verlag, München, 1989.

[114] Philip A. Schrodt. Artificial intelligence and international relations: An overview. In: Valerie M. Hudson, Herausgeber, *Artificial Intelligence and International Politics*, Seiten 9 – 31. Westview Press, Boulder, 1991.

[115] Rudolf Schüßler. Abwanderung, Widerspruch und Kooperation unter Anonymität. In: Matthias Gsänger und Jörg Klawitter, Herausgeber, *Modellbildung und Simulation in den Sozialwissenschaften*, Band 13 von *Forum für interdisziplinäre Forschung*, Seiten 77 – 118. Verlag Dr. Josef H. Röll, Dettelbach, 1995.

[116] Frank L. Sherman. Sherfacs a cross-paradigm, hierachical and contextually sensitive conflict management dataset.
URL=http://www.ai.mit.edu/projects/iiip/ir/sherfacs-overview.htm, 1999.

[117] Wolfgang Stegmüller. *Theorie und Erfahrung*, Band II von *Probleme und Resultate der Wissenschaftstheorie und Analytischen Philosopie*. Springer Verlag, Berlin, Heidelberg, 1986.

[118] Joachim Stender. *Prolog Handbuch*. Hofacker Verlag, Holzkirchen, 1987.

[119] Josef Weingärtner. Eine relationale Prolog–Semantik. Arbeitsbericht 15, Institut für Mathematische Maschinen und Datenverarbeitung, Erlangen, November 1989. Band 22.

[120] Wilhelm Weisweber. *Prolog – Logische Programmierung in der Praxis*. Thomson Publishing, Bonn, erste Auflage, 1997.

[121] Götz Wienold und Christoph Schwarze. Lexical Structure and the Description of Motion Events in Japanese, Korean, Italian and French. Arbeitspapier, FG Sprachwissenschaft, Konstanz, 1989.

[122] Dieter Will. Simulation of crisis based on action types.
URL=http://www.lrz-muenchen.de/∼ua352bm/webserver/webdata/Will.html, 2000.

[123] Niklaus Wirth. Program development by stepwise refinement. *Communications of the ACM*, 14(4):221 – 227, 1971.

[124] Georg Henrik von Wright. Elemente der Handlungslogik. In: Hans Lenk, Herausgeber, *Handlungstheorien – interdisziplinär, Handlungslogik, formale und sprachwissenschaftliche Handlungstheorien*, Band 1 von *Kritische Information, Nr. 62*, Seiten 21 – 34. Wilhelm Fink Verlag, München, 1980.

[125] George Henrik von Wright. *Erklären und Verstehen*. Athenäum Taschenbücher Philosophie. Athenäum Verlag, Königstein/Ts., zweite Auflage, 1984.

[126] Ki Cheon Yoon, Tag Gon Kim und Kyu Ho Park. Discrete event simulation of aircraft mission planning, using DEVSim++. In: Michael J. Chinni, Herausgeber, *Military, Government and Aerospace Simulation*, Seiten 21 – 25, April 1995.

Tabellenverzeichnis

Dateien

Abbildungsverzeichnis

Tabellen

1) verbal:
   (a) Angriffsdrohung,
   (b) Beschuldigen einen Angriff zu planen,
   (c) mit Vergeltungsmaßnahmen drohen,
   (d) mit Sanktionen drohen.
2) physikalisch – politisch:
   (a) politische, soziale und/oder militärische Institutionen in Mißkredit bringen,
   (b) Erschweren von diplomatischen Beziehungen,
   (c) Verletzen von Handelsvereinbarungen,
   (d) Unterdrückung einer ethnischen Minderheit,
   (e) symbolische Verunglimpfung von Werten eines Beteiligten.
3) physikalisch – ökonomisch:
   (a) Embargo,
   (b) Blockade,
   (c) Zurückhalten oder Reduzierung von versprochener ökonomischer Hilfe.
4) physikalisch – gewaltlos militärisch:
   (a) überproportionale Kriegsspiele oder Manöver,
   (b) Mobilmachung,
   (c) Truppenverlegungen näher zur Grenze.
5) physikalisch – gewaltsam militärisch:
   (a) Grenzüberschreitung durch begrenzte Kräfte,
   (b) Invasion des Luftraums,
   (c) Versenken von zivilen und militärischen Schiffen,
   (d) Bombardement von zivilen oder militärischen Zielen,
   (e) großangelegter militärischer Angriff.

Tabelle 12: Dimensionen von Krisen: gerichtete externe feindliche Handlungen nach Michael Brecher, [22], 1977, Seite 64.

1) gewaltfreie militärische Aktionen:
   (a) Waffenproduktion,
   (b) Entwicklung von Waffensystemen (Radarsystem, Nukleare Waffen),
   (c) Bau von Verteidigungsanlagen.
2) politische Aktionen:
   (a) Versuch die Legitimation durch eine internationale Organisation zu erhalten (Beispiel: UNO).
3) verbale interne Aktionen:
   (a) Aufruf zu Ungehorsam durch Medien (Massenmedien, einzelne Medien),
   (b) Unterminierung der Legitimation durch Medien.
4) physikalische Veränderungen für die amtsinhabende Elite:
   (a) Sabotage,
   (b) Terrorismus,
   (c) Demonstrationen,
   (d) Streiks.

Tabelle 13: Dimensionen von Krisen: nichtgerichtete externe Veränderungen
nach Michael Brecher, [22], 1977, Seite 65.

1) verbal,
2) physikalisch,
3) Kombination verbal/physikalisch.

Tabelle 14: Dimensionen von Krisen: Typ der kommunikativen Akte
nach Michael Brecher, [22], 1977, Seite 69.

1) Quelle,
2) Schwere,
3) Komplexität,
4) Intensität,
5) Dauer,
6) Kommunikationsmuster,
7) Ergebnis.

Tabelle 15: Dimensionen von Krisen
nach Michael Brecher, [22], 1977, Seite 63.

1) Systemischer Kontext,
2) geographischer Kontext,
3) Territoriumsgröße,
4) Populationsgröße,
5) Eigenstaatlichkeit,
6) Glaubenssystem (*belief system*),
7) Regimeart,
8) Ökonomie,
9) Militär.

Tabelle 16: Attribute der Akteure
nach Michael Brecher, [22], 1977, Seite 64.

1) Struktur,
2) Größe,
3) bisherige Dauer der Entscheidungsautorität (Amtszeit),
4) Krisenerfahrung,
5) psychologische Umwelt,
6) Informationsprozess,
7) reale und angenommene Optionen.

Tabelle 17: Attribute der Entscheidungseinheit
nach Michael Brecher, [22], 1977, Seite 64.

1) Nummer des Konflikts,
2) Name des Konflikts,
3) Region des Konflikts,
4) Beteiligte,
5) andere Parteien (diplomatische, politische oder wirtschaftliche Unterstützung, Waffenlieferanten),
6) Anzahl der Beteiligten,
7) Initiator,
8) Aggressor,
9) Vermittler,
10) politisches System des Initiators,
11) politische Systeme der Beteiligten,
12) Stufe der wirtschaftlichen und politischen Entwicklung,
13) Anlaß des Konflikts:
    (a) Grenzen,
    (b) nationale Unabhängigkeit,
    (c) Ethische, religiöse oder regionale Autonomie,
    (d) Ideologie,
    (e) internationale Macht und
    (f) Ressourcen.

Tabelle 18: Variablen der Datenbank Kosimo I.

14) Beginn des Konflikts,
15) Ende des Konflikts,
16) Intensität des Konflikts,
17) Instrumente des Konfiktinitiators (Instrumente können Handelsabkommen, Embargos, Sanktionen usw. sein),
18) Instrumente der anderen Parteien,
19) geringste Anzahl der Opfer (die Anzahl der Opfer kann meist nicht genau bestimmt werden. Aus Propagandagründen unterscheiden sich meist die Zahlenangaben der Konfliktparteien.),
20) höchste Anzahl der Opfer,
21) Internationale Konstellation,
22) Reaktion der Nachbarstaten,
23) Reaktion der Supermächte,
24) Einflußsphäre der Supermächte,
25) Konfliktlösungen,
26) Folgen des Konflikts.

Tabelle 19: Variablen der Datenbank Kosimo II. Die Datenbank ist mit dreißig Variablen codiert. Die Diskrepanz zur obigen Anzahl entsteht dadurch, daß manche Variablen mehrfach codiert werden.

esk(miskredit, erschweren).
esk(miskredit, miskredit).
esk(erschweren, handel).
esk(handel, sanktionen).
esk(sanktionen, zurueckhalten).
esk(zurueckhalten, manoever).
esk(manoever, verunglimpfung).
esk(verunglimpfung, aufruf).
esk(aufruf, demonstration).
esk(demonstration, unterminieren).
esk(unterminieren, streik).
esk(streik, mobilmachung).
esk(mobilmachung, truppenverlegung).
esk(truppenverlegung, embargo).
esk(embargo, waffenproduktion).
esk(waffenproduktion, konstruktion).
esk(konstruktion, anlagenbau).
esk(anlagenbau, beschuldigung).
esk(beschuldigung, angriffsdrohung).
esk(angriffsdrohung, vergeltung).
esk(vergeltung, legitimation).
esk(legitimation, sabotage).
esk(sabotage, unterdrueckung).
esk(unterdrueckung, terror).
esk(terror, blockade).
esk(blockade, grenzueberschreitung).
esk(grenzueberschreitung, luftraum).
esk(luftraum, schiffeversenken).
esk(schiffeversenken, bombardieren).
esk(bombardieren, grossangriff).
esk(skip, miskredit).
esk(skip, skip).
esk(grossangriff, grossangriff).
esk(aufgeben, aufgeben).

Tabelle 20: Eskalation der Handlungen.

% Gruppe 1: verbal

gr(angriffsdrohung, beschuldigung).
gr(angriffsdrohung, angriffsdrohung).
gr(beschuldigung, vergeltung).
gr(vergeltung, sanktionen).
gr(sanktionen, sanktionen).

% Gruppe 2: physikalisch – politisch

gr(miskredit, erschweren).
gr(miskredit, miskredit).
gr(erschweren, handel).
gr(handel, unterdrueckung).
gr(unterdrueckung, verunglimpfung).
gr(verunglimpfung, verunglimpfung).

% Gruppe 3: physikalisch – ökonomisch

gr(embargo, blockade).
gr(embargo, embargo).
gr(blockade, zurueckhalten).
gr(zurueckhalten, zurueckhalten).

% Gruppe 4: physikalisch – gewaltlos militärisch

gr(manoever, mobilmachung).
gr(manoever, manoever).
gr(mobilmachung, truppenverlegung).
gr(truppenverlegung, truppenverlegung).

% Gruppe 5: physikalisch – gewaltsam militärisch

gr(grenzueberschreitung, luftraum).
gr(grenzueberschreitung, grenzueberschreitung).
gr(luftraum, schiffeversenken).
gr(schiffeversenken, bombardieren).
gr(bombardieren, grossangriff).
gr(grossangriff, grossangriff).

Tabelle 21: Eskalation der Handlungen in Gruppen I von II.

% Gruppe 6: gewaltfreie militärische Aktion

gr(waffenproduktion,konstruktion).
gr(waffenproduktion,waffenproduktion).
gr(konstruktion,anlagenbau).
gr(anlagenbau,anlagenbau).

% Gruppe 7: politische Aktion

gr(legitimation,legitimation).

% Gruppe 8: verbale interne Aktion

gr(aufruf,unterminieren).
gr(aufruf,aufruf).
gr(unterminieren,unterminieren).

% Gruppe 9: physikalische Veränderung für die amtsinhabende Elite

gr(sabotage,terror).
gr(sabotage,sabotage).
gr(terror,demonstration).
gr(demonstration,streik).
gr(streik,streik).

% Gruppe 10: skip und aufgeben

gr(skip,miskredit).
gr(skip,skip).
gr(aufgeben,aufgeben).

Tabelle 22: Eskalation der Handlungen in Gruppen II von II.

```
:-op(700,xfx,[<~,~>,.<,.>]).
% Umkehrrelation
Y ~> X :- X <~ Y.
% Transitivitaet
X.<Y :-
  X <~ Y.

X.<Y :-
  X <~ Z,Z.<Y.

X.<X.

X.>Y :-Y.<X.

verbal <~ pyspol.
pyspol <~ oeko.
oeko <~ gfmil.
gfmil <~ mil.

gfmil <~ host.
host <~ mil.

angriffsdrohung <~ angriffsbeschuldigung.
angriffsbeschuldigung <~ vergeltungsdrohung.
vergeltungsdrohung <~ sanktionssdrohung.

miskredit <~ erschweren_diplom_bez.
erschweren_diplom_bez <~ verletzung.
verletzung <~ unterdrueckung.
unterdrueckung <~ verunglimpfung.

⋮

grenzueberschreitung <~ luftraum.
luftraum <~ schiffeversenken.
schiffeversenken <~ bombardement.
bombardement <~ angriff.

⋮
```

Tabelle 23: Eskalation der Handlungen in Gruppen III.

Programmdateien:			
Funktion	Name	Zeilen	Seite
Hauptprogramm	prog.prolog	814	XXX
Hilfsprädikate	pred.prolog	262	XLIX
Menüprogramm	monitor.prolog	711	LV
Kontrolldatei	feasable.prolog	18	LXXI
Datei der Zähler	flags.prolog	202	LXXII
Datendateien:			
Funktion	Name	Zeilen	Seite
Parameterdatei	para.prolog	21	LXXVII
Datei der Handlungen	actions.prolog	45	LXXVIII
aktuelle Charaktere	charakter.prolog	31	LXXIX
Ausnahmebehandlung	cha.prolog	31	LXXIX
Charakterliste	charakters.prolog	20	XXX
Charakterliste	glchs.prolog	21	LXXXI
Charakterliste	92chs.prolog	21	LXXXII
Regeldateien:			
Funktion	Name	Zeilen	Seite
Verzeichnis	rules.prolog	28	LXXXIII
Eskalation	linear	83	LXXXIV
Eskalation nach Gruppen	lingr	101	LXXXVI
verschärfte Eskalation	lin2	25	LXXXIX
Deeskalation	desk	41	XC
Deeskalation nach Gruppen	deskgr	41	XCI
verschärfte Deeskalation	desk2	25	XCII
gleiche Handlung	titfortat	26	XCIII
gleicher Handlungstyp	tfts	24	XCIV
Großangriff	maxesk	18	XCV

Tabelle 24: Liste der Dateien I

Statistik:			
Funktion	Name	Zeilen	Seite
Statistik	stat.prolog	193	XCVI
Ergebnisdateien:			
Funktion	Name	Anzahl Zeilen	Seite
Ergebnis	Ergebnis	ca. 1.800	CI
Ende	ende.prolog	runs $\times 3$	CII
Geschichte	history.prolog	$(30 + \mathit{Perioden}) \times \mathit{runs}$	CIII
Handlungstypen	tabellegesamt	Handlungstypen $+2$	CIV
Bildschirmprotokoll	protokoll.sve	$\mathit{Charaktere} \times 2 + 100$	CXII
Handlungstypen	ergebnis_neu	Anzahl Charaktere	CV
Periodenende	maxende.prolog	bis 10.000	CVI

Tabelle 25: Liste der Dateien II

Dateien

Datei 1: prog.prolog

```
/*********************************************/
/***            Programm zur Krisensimulation      ***/
/*********************************************/
%
% Dieses File ist das Hauptprogramm zur Krisensimulation:
%
% Aufruf erfolgt mit: 'start.'
% start/0
%
% letzte Aenderung: 00/3/20
%
% filename: prog.prolog
%
/*********************************************/
/***                          start/0                ***/
/*********************************************/
%
% start/0
%
% unterstuetzt Statistiken, laedt benoetigte Dateien ein,
% und startet begin/0
%
start:-
  get_time(X),
  statistics(cputime,CPUOLDTIME),
  statistics(inferences,Inferences),
  protocol('protokoll.txt'),
  style_check([-singleton]),
  ['pred.prolog','para.prolog',
  'actions.prolog','feasable.prolog','rules.prolog',
  'flags.prolog'],
  consult_facts,
  del('history.prolog'),
  del('ende.prolog'),
  asserta(run(1)),
  use_old_data(Yes_No),
  (Yes_No = no,
  initsialisierung(1)
```

```
45 ;
46   Yes_No = yes,
47   get_old_data(1)),
48   begin,
49   statistik(CPUOLDTIME,Inferences),
50   get_time(Y),
51   write_time(X,Y),
52   put(7),
53   protocolwrite,
54   noprotocol.
55
56 /*****************************************/
58 /***                    consult_facts/0          ***/
60 /*****************************************/
62 %
63 % consult_facts/0
64 % benoetigt: number_actions/0
65 %       rules_ini/0
66 %
67 % Laedt Regeln und Handlungsanzahl
68 %
69 consult_facts :-
70   rules_ini,
71   number_actions.
72
73 number_actions :-
74   findall(Nummer,act(_,_,_,Nummer),Anzahl),
75   length(Anzahl,Laenge),
76   (retract(handlungsanzahl(_)),fail
77 ; true),
78   asserta(handlungsanzahl(Laenge)).
79
80 rules_ini :-
81   findall(Art,rule(Art),Rules),
82   asserta(rule_list(Rules)),
83   length(Rules,Laenge),
84   (between(1,Laenge,X),
85   nth1(X,Rules,Name),
86   consult(Name),
87   fail
88 ; true).
```

```
89
90 /*****************************************/
92 /***                 initsialisierung/1            ***/
94 /*****************************************/
96 %
97 % initsialisierung/1
98 %
99 % Initsialisierung von Charakteren und erster Handlung,
100 % falls keine alten Daten verwendet werden.
101 %
102 initsialisierung(R):-
103   del('charakter.prolog'),
104   (retract(charakter(_,_,_,_)),fail
105 ; true),
106   created_actors,
107   retractall(run(_)),
108   asserta(run(R)),
109   retractall(period(_)),
110   asserta(period(0)),
111   choosed_actor(AS),
112   handlungsanzahl(Laenge),
113   %'erste Handlung darf nicht skip oder aufgeben sein'
114   Random is Laenge-2,
115   Handlungsnummer is random(Random)+1,
116   act(Handlungsname,1,Maxstaerke,Handlungsnummer),
117   Neuestaerke is random(Maxstaerke)+1,
118   direktion_ini,
119   direktion(Direktion),
120   asserta(action(R,0,AS,Direktion,Handlungsname,
121             Neuestaerke,Maxstaerke,Handlungsnummer)),
122   count_flag(Handlungsname,Direktion),
123   write_initial(action(R,0,AS,Direktion,Handlungsname,
124             Neuestaerke,Maxstaerke,Handlungsnummer)),!.
125
126 /*****************************************/
128 /***                 get_old_data/1            ***/
130 /*****************************************/
132 %
133 % get_old_data/1
134 % verwendet: exists_charakter/0
135 %       charakterfile_exists/0
```

```
136 %       initial_old/1
137 %       direction_ini/0
138 %       choosed_actor/1
139 %       created_actors/0
140 %       charakter_rule/2
141 %       fanatiker/1
142 %
143 % Initialisierung von Charakteren und erster Handlung,
144 % falls alte Daten verwendet werden.
145
146 get_old_data(R):-
147   exists_charakter,
148   asserta(run(R)),
149   asserta(period(0)),
150   actors(AS),
151   rule_list(Rules),
152   (between(1,AS,Nummerakteur),
153   charakter(Nummerakteur,Gewichtung,Strength,Fanatiker),
154   append('history.prolog'),
155   write_charakter(Nummerakteur,Gewichtung,Rules,
156                 Strength,Fanatiker),
157   told,
158   fail
159   ;true),
160   initial_old(R).
161
162 exists_charakter:-
163   (actors(AS),
164   (between(1,AS,Nummerakteur),
165   charakter(Nummerakteur,Gewichtung,Strength,Fanatiker)))
166 ;
167   charakterfile_exist,/*Ausnahmebehandlung falls Datei
168         'charakter.prolognicht existiert und gleichzeitig
169         alte Datenverwendet werden sollen*/
170   consult('charakter.prolog').
171
172 charakterfile_exist:-
173   exists_file('charakter.prolog')
174 ;
175   append("cpcha.prolog","charakter.prolog",C),
176   name(Kommando,C),
```

```
177   shell(Kommando).
178
179 initial_old(R):-
180   choosed_actor(AS),
181   number_actions,
182   handlungsanzahl(Laenge),
183   %'erste Handlung darf nicht skip oder aufgeben sein'
184   Random is Laenge-2,
185   Handlungsnummer is random(Random)+1,
186   act(Handlungsname,Staerke,Maxstaerke,Handlungsnummer),
187   Neuestaerke is random(Maxstaerke)+1,
188   direktion_ini,
189   direktion(Direktion),
190   asserta(action(R,0,AS,Direktion,Handlungsname,
191             Neuestaerke,Maxstaerke,Handlungsnummer)),
192   count_flag(Handlungsname,Direktion),
193   write_initial(action(R,0,AS,Direktion,Handlungsname,
194             Neuestaerke,Maxstaerke,Handlungsnummer)),!.
195
196 direktion_ini:-
197   (retract(direktion(X)),fail
198 ;true),
199   X is 1,
200 % Falls mehrere Arten der Nichtausfuehrung
201 % von Handlungen gewuenscht werden:
202 % X is random(3)+1
203   nth1(X,[positiv,negativ,unterlassung],Direktion),
204   asserta(direktion(Direktion)).
205
206 choosed_actor(AS):-
207   actor_list(L),
208   length(L,E),
209   Y is random(E)+1,
210   nth1(Y,L,AS),
211   (retract(active_actor(_)),fail
212 ;true),
213   asserta(active_actor(AS)).
214
215 created_actors:-
216   rule_list(Rules),
217   actors(AS),
```

```
218   findall(I,between(1,AS,I),L),
219   asserta(actor_list(L)),
220   append('charakter.prolog'),
221   write('actor_list('),write(L),write(').'),nl,nl,told,
222   (between(1,AS,N),
223   charakter_rule(N,Rules),
224   fail
225 ; true).
226
227 charakter_rule(Akteur,Rules):-
228   maplist(rulenumber,Rules,X),
229   calculate_sum(X,S),
230   weigth(X,S,Gewichtung),
231   Strength is random(100)+1,
232   fanatiker(Yes_or_no),
233   asserta(charakter(Akteur,Gewichtung,
234                  Strength,Yes_or_no)),
235   append('charakter.prolog'),
236   write_charakter(Akteur,Gewichtung,Rules,
237                  Strength,Yes_or_no),
238   told,
239   append('history.prolog'),
240   write_charakter(Akteur,Gewichtung,Rules,
241                  Strength,Yes_or_no),
242   told.
243
244 fanatiker(Yes_or_no):-
245   Zufallszahl is random(2)+1,
246   nth1(Zufallszahl,[yes,no],Yes_or_no).
247
248 /*****************************************/
250 /***                    Hauptschleifen          ***/
252 /*****************************************/
254 %
255 % begin/0
256 % mainloop/3
257 %
258 % startet die Schleifen
259 %
260 begin:-
261   runs(RR),
```

```
262   periods(TT),
263   TT1 is TT+1,
264   (between(1,RR,R),mainloop(R,TT1,RR),
265   fail
266 ; true
267   ),!.
268
269 mainloop(R,TT1,RR):-
270   (retract(period(_)),fail
271 ; true),
272   asserta(period(1)),
273   flag(period,_,1),!,
274   repeat,
275   flag(period,T,T+1),
276 % write(T),tab(1),write(R),nl,
277   tabelle_1(TT1,T,R),
278   TT1<T,
279   !,adjust_mainloop(R,RR).
280
281 /*****************************************/
283 /***                                   ***/
285 /***          Entscheidungstabelle     ***/
287 /***                                   ***/
289 /*****************************************/
291 %
292 %
293 % Aufruf erfolgt mit: 'tabelle_1(TT1,T,R)'
294 %
295 /*****************************************/
297 /***            tabelle_1/3            ***/
299 /*****************************************/
301 %
302 % tabelle_1/3
303 %
304 % tabelle_1 ist eine Entscheidungstabelle fuer die
305 % Perioden eines Laufs. Dabei wird entweder ein Akteur
306 % ausgewaehlt und eine Handlung ausgefuehrt oder der
307 % Abbruch der Periode herbeigefuehrt.
308 %
309 % Ende, da vorherige Handlung 'grossangriff' ist.
310 %
```

```
311 tabelle_1(TT,T,R):-
312   vorher_contr(T,R),
313   !,
314   counter_up(TT,T),
315   flag(ende_grossangriff,G,G+1),
316   ende(R,T),
317   end(R,T).
318
319 % Ende, da vorherige Handlung 'aufgeben' ist.
320
321 tabelle_1(TT,T,R):-
322   vorher_aufg(T,R),
323   !,
324   counter_up(TT,T),
325   flag(ende_aufgeben,A,A+1),
326   ende(R,T),
327   end(R,T).
328
329 % Ende, da der Gegner viel staerker ist.
330
331 tabelle_1(TT,T,R):-
332   (TT<T,true,!)
333 ;(
334   T>50,
335   selected_actors(AS1,AS2),
336   once(charakter(AS1,Gewichtung1,St1,no)),
337   once(charakter(AS2,Gewichtung2,St2,_)),
338   ST is St2+95,
339   St1<ST,!,
340   kernel_aufgeben(AS1,R,T),
341   counter_up(TT,T),
342   T1 is T+1,
343   flag(ende_aufgeben,S,S+1),
344   ende(R,T1),
345   end(R,T1)
346   ),
347   !.
348
349 % Ende, da Gegner staerker und erheblich eskaliert
350
351 tabelle_1(TT,T,R):-
```

```
352   (TT < T,true,!)
353 ; (
354   T > 50,
355   selected_actors(AS1,AS2),
356   once(charakter(AS1,Gewichtung1,St1,no)),
357   once(charakter(AS2,Gewichtung2,St2,_)),
358   ST is St2+95,
359   St1 < ST,
360   Vor is T-1,
361   action(R,Vor,AS2,positiv,Handlungsname,_,_,_),
362   action(R,_,AS1,_,Handlung,_,_,_),
363   esk(Handlung,H3),esk(H3,Handlungsname),
364   !,
365   kernel_aufgeben(AS1,R,T),
366   counter_up(TT,T),
367   T1 is T+1,
368   flag(ende_aufgeben,S,S+1),
369   ende(R,T1),
370   end(R,T1)
371   ),
372   !.
373
374 % Ende, da Zaehler heraufgesetzt wurde.
375
376 tabelle_1(TT,T,R):-
377   TT < T,
378   !.
379
380 % Ende, da maximale Anzahl der Perioden erreicht ist.
381
382 tabelle_1(TT,T,R):-
383   TT = T,
384   !,
385   flag(ende_periodsmax,K,K+1),
386   ende_normal(R,T,TT).
387
388 % normaler Verlauf
389
390 tabelle_1(_,T,R):-
391   !,
392   kernel(R,T).
```

```
393 /*******************************************/
395 /***        Kontrolle der vorhergehenden Handlung   ***/
397 /*******************************************/
399 %
400 % vorher_contr/2
401 % Kontrolliert ob die vorherige Handlung 'grossangriff' ist.
402 %
403 % vorher_aufg/2
404 % Kontrolliert ob die vorherige Handlung 'aufgeben' ist.
405 %
406 % Bei Bedarf koennen weitere Handlungen, die einen Abbruch
407 % der Verabeitung herbeifuehren, eingefuegt werden.
408 %
409 vorher_contr(T,R):-
410   Vorher is T-1,
411   once(action(R,Vorher,_,Direktion,Handlungsname,_,_,_)),
412   once(member(Handlungsname,[grossangriff])),
413   once(Direktion=positiv),!.
414
415 vorher_aufg(T,R):-
416   Vorher is T-1,
417   once(action(R,Vorher,_,Direktion,Handlungsname,_,_,_)),
418   once(member(Handlungsname,[aufgeben])),
419   once(Direktion=positiv),!.
420
421 /*******************************************/
423 /***                    counter_up/2                ***/
425 /*******************************************/
427 %
428 % counter_up/2
429 %
430 % Setzt den Periodenzaehler groesser als die Anzahl der
431 % maximalen Perioden und fuehrt
432 % so einen Abbruch
433 % der repeat---Schleife herbei.
434 %
435 counter_up(TT,T):-
436   V is TT+5,
437   flag(period,_,V).
438
439 /*******************************************/
```

```
441 /***                    selected_actors/2            ***/
443 /*****************************************************/
445 %
446 % selected_actors/2
447 %
448 % waehlt zwei Akteure aus.
449 %
450 selected_actors(AS1,AS2):-
451   actor_list(L),
452   length(L,E),
453   E > 1,
454   Y is random(E)+1,
455   nth1(Y,L,AS1),
456   delete(L,AS1,L_new),
457   length(L_new,F),
458   X is random(F)+1,
459   nth1(X,L_new,AS2).
460
461 /*****************************************************/
463 /***                    adjust_mainloop/2             ***/
465 /*****************************************************/
467 %
468 % adjust_mainloop/2
469 %
470 % Setzt den Rundenzaehler um eins hoeher,
471 % und begint neue Runde.
472 %
473 adjust_mainloop(R,RR):-
474   R1 is R+1,
475   retract(run(R)),
476   asserta(run(R1)),
477   R < RR,
478   append('history.prolog'),
479   nl,nl,nl,write('%'),tab(1),
480   write('-------------------------'),nl,
481   write('%'),tab(1),
482   write('-------------------------'),tab(1),
483   write('neue'),tab(1),write('Runde'),tab(1),
484   write('-------------------------'),nl,
485   write('%'),tab(1),
486   write('-------------------------'),nl,nl,
```

```
487   told,
488   entferne_teil,
489   use_old_data(X),
490   (X = no,
491   initsialisierung(R1)
492 ;
493   X = yes,
494   get_old_data(R1)),
495   !.
496
497 /*****************************************/
499 /***                Kernel             ***/
501 /*****************************************/
503 %
504 % kernel_aufgeben/3
505 % kernel/2
506 %
507 % fuehrt die mit der Entscheidungstabelle ausgewaehlte
508 % Handlung aus und erhoeht mit adjust/1 die Anzahl
509 % der Perioden um 1.
510 %
511 kernel_aufgeben(AS,R,T):-
512   asserta(action(R,T,AS,positiv,aufgeben,1,1,32)),
513   (retract(choosed_actionart),fail
514 ; true),
515   asserta(choosed_actionart(act(aufgeben,1,1,32))),
516   (retract(used_rule),fail
517 ; true),
518   asserta(used_rule(aufgeben)),
519   (retract(direktion),fail
520 ; true),
521   asserta(direktion(positiv)),
522   count_flag(aufgeben,positiv),
523   write_action(R,T,aufgeben,aufgeben,positiv,1,1,32,AS),
524   adjust(T),
525   !.
526
527 kernel(R,T):-
528   choosed_actor(AS),
529   active_actor(AS),
530   consult_profile(AS,M),
```

```
531   used_rule(M),
532   choosed_action(R,T,M,AS,Staerke,Nr),
533   choosed_actionart(act(H,1,Max,Num)),
534   direktion(Direktion),
535   action(R,T,AS,Direktion,Handlung,Neustaerke,X,Num),
536   count_flag(H,Direktion),
537   write_action(R,T,M,H,Direktion,Neustaerke,Max,Num,AS),
538   adjust(T),!.
539
540 adjust(T):-
541   T1 is T+1,
542   retract(period(T)),
543   asserta(period(T1)).
544
545 /*********************************************/
547 /***                  consult_profile/2          ***/
549 /*********************************************/
551 %
552 % consult_profile/2
553 % waehlt Handlungsregel, abhaengig vom Charakter
554 %
555 consult_profile(AS,M):-
556   charakter(AS,Liste,Strength,Fanatiker),
557   Zufallszahl is random(100)+1,
558   length(Liste,Zahl),
559   rule_list(Rules),
560   asserta(aux_sum(0)),
561   (between(1,Zahl,Auswahl),
562   add(Auswahl,Zufallszahl,Liste,Hilfssumme),
563   Zufallszahl =< Hilfssumme),
564   nth1(Auswahl,Rules,M),
565   asserta(used_rule(M)),
566   retract(aux_sum(SS)),!.
567
568 /*********************************************/
570 /***                    Hilfspraedikate          ***/
572 /*********************************************/
574 %
575 % add/4
576 % choosed_action/6
577 %
```

```
578 add(Auswahl,Zufallszahl,Liste,Hilfssumme):-
579   aux_sum(Alte_hilfssumme),
580   nth1(Auswahl,Liste,Zwischensumme),
581   Hilfssumme is Alte_hilfssumme+Zwischensumme,
582   retract(aux_sum(Alte_hilfssumme)),
583   asserta(aux_sum(Hilfssumme)),!.
584
585 choosed_action(R,T,M,AS,Staerke,Nr):-
586   used_rule(M),
587   call(M,R,T,AS,Staerke,Nr),!.
588
589 /*****************************************/
591 /***                Datenbasis bereinigen          ***/
593 /*****************************************/
595 %
596 % entferne_teil/0
597 % entferne_alle/0
598 %
599 % loescht alle Handlungen (action)
600 % in der Datenbasis des Programms
601 %
602 entferne_teil:-
603   retract(action(_,_,_,_,_,_,_,_)),fail
604 ; true,
605   garbage_collect,
606   trim_stacks.
607
608 entferne_alle:-
609   retract(action(_,_,_,_,_,_,_,_)),fail
610 ; true,
611   retract(used_rule(_)),fail
612 ; true,
613   retract(rule_list(_)),fail
614 ; true,
615   retract(choosed_action(act(_,_,_,_))),fail
616 ; true,
617 % retract((_)),fail
618 ; true,
619 % retract(period(_)),fail
620 ; true
621 ;
```

```
622   garbage_collect,
623   trim_stacks.
624
625 /*****************************************/
627 /***          schreiben in Dateien       ***/
629 /*****************************************/
631 %
632 % write_initial/1
633 % write_action/9
634 % write_charakter/5
635 % end/2
636 % ende/2
637 % ende_normal/3
638 % write_max/3
639 %
640 write_initial(action(R,0,AS,Direktion,Handlungsname,
641             Neuestaerke,Maxstaerke,Handlungsnummer)):-
642   append('history.prolog'),
643   write('%'),tab(1),
644   write('action(Run,Periode,Nummerakteur,Direktion,'),nl,
645   write('%'),
646   tab(15),
647   write('Handlungsname,Neuestaerke,Maxstaerke,'),nl,
648   write('%'),tab(45),write('Handlungsnummer).'),nl,nl,
649   write('%'),tab(1),
650   write('Initsialisierungsperiode 0 des Runs'),
651   tab(1),write(R),tab(1),
652   write('--'),
653   write('noch keine Regel benutzt'),nl,
654   write(action(R,0,AS,Direktion,Handlungsname,
655             Neuestaerke,Maxstaerke,Handlungsnummer)),
656   write('.'),
657   tab(3),
658   nl,
659   told,!.
660
661 write_action(R,T,M,Handlungsname,Direktion,
662                   Neustaerke,Max,Nr,AS):-
663   choosed_actionart(act(Handlungsname,1,Max,Nr)),
664   used_rule(M),
665   act(Handlungsname,1,Maxstaerke,Nr),
```

```
666   Nummerakteur is AS,
667   period(T),
668   run(R),
669   direktion(Direktion),
670   asserta(action(R,T,Nummerakteur,Direktion,Handlungsname,
671                  Neustaerke,Maxstaerke,Nr)),
672   append('history.prolog'),
673   write('%'),tab(1),
674   write('Run :'),tab(1),write(R),tab(2),
675   write('Periode :'),tab(1),write(T),
676   tab(2),write('Akteur :'),tab(1),write(AS),
677   tab(3),
678   write('verwendete Regel :'),tab(1),
679   write(M),
680   nl,
681   write(action(R,T,Nummerakteur,Direktion,Handlungsname,
682       Neustaerke,Maxstaerke,Nr)),
683   write('.'),
684   nl,
685   told,!.
686
687 write_charakter(Akteur,Gewichtung,Rules,Strength,Fanatiker):-
688   write('%'),tab(1),
689   write('----------------------------------------'),
690   tab(2),
691   write('Akteur'),tab(1),write(Akteur),
692   tab(2),
693   write('----------------------------------------'),
694   nl,
695   write('%'),tab(1),write('Regeln :'),write(Rules),nl,
696   write('%'),tab(1),write('Staerke :'),write(Strength),
697   tab(2),write('Fanatiker :'),write(Fanatiker),
698   nl,
699   write('charakter'),write('('),
700   write(Akteur),write(','),
701   write(Gewichtung),write(','),write(Strength),
702   write(','),write(Fanatiker),
703   write(')'),write('.'),
704   nl,nl.
705
706 end(R,T):-
```

```
707   append('history.prolog'),
708   nl,
709   write('%'),tab(1),
710   write('-------------------------------------'),nl,
711   write('%'),tab(1),
712   write('Der Run'),tab(1),write(R),tab(1),write('ist beendet'),nl,
713   write('%'),tab(1),
714   Vorher is T-1,
715   write('Ende bei Periode:'),tab(1),write(Vorher),nl,
716   write('%'),tab(1),
717   write('-------------------------------------'),nl,
718   told,!.
719
720 ende(R,T):-
721   Vorher is T-1,
722   used_rule(M),
723   action(R,Vorher,AS,Direktion,Handlungsname,
724                        Neustaerke,Maxstaerke,Handlungsnummer),
725   append('ende.prolog'),
726   write('%'),tab(1),
727   write('Der Run'),tab(1),write(R),tab(1),write('ist beendet.'),tab(1),
728   write('Ende bei Periode:'),tab(1),write(Vorher),nl,
729   write('%'),tab(1),write('Akteur:'),tab(1),write(AS),
730   tab(3),
731   write('letzte verwendete Regel:'),tab(1),
732   write(M),
733   nl,
734   write(action(R,Vorher,AS,Direktion,Handlungsname,
735                        Neustaerke,Maxstaerke,Handlungsnummer)),
736   write('.'),
737   nl,
738   told,!.
739
740 ende_normal(R,T,TT):-
741   Vorher is TT-1,
742   used_rule(M),
743   action(R,Vorher,AS,Direktion,Handlungsname,
744                        Neustaerke,Maxstaerke,Handlungsnummer),
745   append('ende.prolog'),
746   write('%'),tab(1),
747   write('Der Run'),tab(1),write(R),tab(1),
```

```
748   write('wurde abgebrochen.'),tab(1),nl,
749   write('%'),tab(1),
750   write('Ende bei Periode:'),tab(1),write(Vorher),
751   tab(1),write('----'),tab(1),
752   write('Maximale Anzahl der Perioden erreicht.'),nl,
753   write('%'),tab(1),write('Akteur:'),tab(1),write(AS),
754   tab(3),
755   write('letzte verwendete Regel:'),tab(1),
756   write(M),
757   nl,
758   write('%'),tab(1),write('letzte Handlung:'),nl,
759   write(action(R,Vorher,AS,Direktion,Handlungsname,
760                    Neustaerke,Maxstaerke,Handlungsnummer)),
761   write('.'),
762   nl,
763   told,!,
764   write_max(R,TT,T).
765
766 write_max(R,TT,T):-
767   rule_list(Rules),
768   selected_actors(AS1,AS2),
769   once(charakter(AS1,Gewichtung1,St1,Y)),
770   once(charakter(AS2,Gewichtung2,St2,N)),
771   Vorher is TT-1,
772   used_rule(M),
773   action(R,Vorher,AS,Direktion,Handlungsname,
774                    Neustaerke,Maxstaerke,Handlungsnummer),
775   append('maxende.prolog'),
776   write('%'),tab(1),
777   write('Der Run'),tab(1),write(R),tab(1),
778   write('wurde abgebrochen.'),tab(1),nl,
779   write('%'),tab(1),
780   write('Ende bei Periode:'),tab(1),
781   write(Vorher),tab(1),write('----'),tab(1),
782   write('Maximale Anzahl der Perioden erreicht.'),nl,
783   write('%'),tab(1),write('Akteur:'),tab(1),write(AS),
784   tab(3),
785   write('letzte verwendete Regel:'),tab(1),
786   write(M),
787   nl,
788   write('%'),tab(1),write('letzte Handlung:'),nl,
```

```
789   write(action(R,Vorher,AS,Direktion,Handlungsname,
790                    Neustaerke,Maxstaerke,Handlungsnummer)),
791   write('.'),
792   nl,
793   write('CharakterderAkteure :'),nl,
794   write_charakter(AS1,Gewichtung1,Rules,St1,Y),nl,
795   write_charakter(AS2,Gewichtung2,Rules,St2,N),nl,
796   told,!.
797
798 /*****************************************/
800 /***                    Todo--Liste          ***/
802 /*****************************************/
804 %
805 % neue Regeln
806 %
807 /*****************************************/
809 /***                    End of file          ***/
811 /*****************************************/
813 %
814 % End of file prog.prolog
```

Datei 2: pred.prolog

```
1  /*********************************************/
3  /***                    Praedikatmodul          ***/
5  /*********************************************/
7  %
8  %
9  % letzte Aenderung: 10/2/00
10 %
11 % Aufruf erfolgt durch 'consult/1'
12 %
13 % Dateiname: pred.prolog
14 %
15 /*********************************************/
17 /***                      del/1                ***/
19 /*********************************************/
21 %
22 % del/1
23 %
24 % loescht Datei X, falls sie existiert.
25 %
26 del(X):-
27   exists_file(X)->
28   delete_file(X)
29 ;true.
31 /*********************************************/
33 /***                   calculate_sum/2         ***/
35 /*********************************************/
37 %
38 % calculate_sum/2
39 %
40 % summiert eine Liste L von Integers
41 %
42 calculate_sum(L,S):-
43   asserta(counter(0)),
44   length(L,E),
45   (between(1,E,X),
46   auxpred(L,X),
47   fail
48 ;true),
49   counter(S),
```

```
50   retract(counter(S)).
51
52 auxpred(L,X):-
53   nth1(X,L,N),
54   counter(C),
55   C1 is C+N,
56   retract(counter(C)),
57   asserta(counter(C1)),!.
58
59 /*********************************************/
61 /***                    rulenumber/2                  ***/
63 /*********************************************/
65 %
66 % rulenumber/2
67 %
68 %erstellt fuer alle Elemente einer Liste eine Zufallszahl.
69 % Aufruf erfolgt innnerhalb des Systempraedikats maplist/3.
70 %
71 rulenumber(Rules,X):-
72   X is random(100)+1.
73
74 /*********************************************/
76 /***                      weigth/3                    ***/
78 /*********************************************/
80 %
81 % weigth/3
82 %
83 % kalibriert die Gewichte auf 100
84 %
85 weigth(Liste,Summe,Ergebnissliste):-
86   asserta(list([])),
87   length(Liste,E),
88   (between(1,E,Zaehler),
89   weigth_auxpred(Liste,Zaehler,Summe,Gewichtung),
90   fail
91 ;true),
92   list(Gewichtung),
93   reverse(Gewichtung,Ergebnissliste),
94   retract(list(Gewichtung)),
95   is_list(Ergebnissliste).
96
```

```
97  weigth_auxpred(Liste,Zaehler,Summe,Gewichtung):-
98    nth1(Zaehler,Liste,Element),
99    list(Hilfsliste),
100   Z is (Element*100)/Summe,
101   append([Z],Hilfsliste,Hilfsliste1),
102   retract(list(Hilfsliste)),
103   asserta(list(Hilfsliste1)),
104   !.
105
106 /*********************************************/
108 /***                    tabellenzeile/4          ***/
110 /*********************************************/
112 %
113 % tabellenzeile/4
114 %
115 tabellenzeile(TabellenName,SpaltenName,Inhalt,TabellenZeile):-
116   tabelle(Ueberschrift),
117   functor(Ueberschrift,TabellenName,SpaltenZahl),
118   arg(Spalte,Ueberschrift,SpaltenName),
119   functor(TabellenZeile,TabellenName,SpaltenZahl),
120   arg(Spalte,TabellenZeile,Inhalt),
121   clause(TabellenZeile,_).
122
123 /*********************************************/
125 /***                    spalteninhalt/3          ***/
127 /*********************************************/
129 %
130 % spalteninhalt/3
131 %
132 spalteninhalt(TabellenName,SpaltenName,Inhalt):-
133   tabellenzeile(TabellenName,SpaltenName,Inhalt,_).
134
135 /*********************************************/
137 /***                      Mengen               ***/
139 /*********************************************/
141 %
142 % kart_prod/3
143 % paare/3
144 % permut/2
145 %
146
```

```
147 kart_prod([ ],_,[ ]).
148
149 kart_prod([X | R1],L,L3) :-
150   paare(X,L,L1),
151   kart_prod(R1,L,L2),
152   append(L1,L2,L3).
153
154 /* paare/3 ist natuerlich auch restrekursiv. */
155
156 paare(_,[ ],[ ]).
157
158 paare(X,[Y | R],[[X,Y] | R1]) :-
159   paare(X,R,R1).
160
161 /* Aufruf: kart_prod([yes,no],[no,yes],X). */
162
163 sortiere(Liste,Rliste) :-
164   sort_akku(Liste,[ ],Rliste).
165
166 sort_akku([ ],Akku,Akku).
167
168 sort_akku([[Head | Tail] | Rest],Akku,Rliste) :-
169   (Head = Tail,true
170 ;
171   msort([Head | Tail],Element1)),
172   sort_akku(Rest,[Element1 | Akku],Rliste).
173
174 /* Aufruf: kart_prod([a,b,c],Y). */
175
176 permut(X,Z) :-
177   kart_prod(X,X,A),
178   sortiere(A,B),
179   sort(B,Z).
180
181 /*****************************************/
183 /***                protokolwrite/0        ***/
185 /*****************************************/
187 %
188 % protocolwrite/0
189 %
190 protocolwrite :-
```

```
191   (exists_file('protokoll.txt'),
192   shell('mvprotokoll.txtprotokoll.sve'))
193 ; true.
194
195 /*********************************************/
197 /***                    statistik/0          ***/
199 /*********************************************/
201 %
202 % statistik/0
203 %
204 statistik(CPUOLDTIME, Inferences) :-
205   nl,nl,
206   statistics,
207   statistics(cputime, CPUTIME),
208   Cpu is CPUTIME - CPUOLDTIME,
209   statistics(inferences, Infer),
210   Inf is Infer - Inferences,
211   nl,tab(5),
212   write('--------'),tab(2),
213   write('CPU-TIMEist'),tab(1),
214   write(Cpu),tab(1),write('Sekunden'),tab(2),
215   write('--------'),
216   nl,tab(5),
217   write('--------'),tab(2),
218   write('CPU-TIMEist'),tab(1),
219   Cp is Cpu/60,
220   write(Cp),tab(1),write('Minuten'),tab(2),write('--'),
221   nl,tab(8),
222   write('-----'),tab(2),
223   write('for'),tab(2),
224   write(Inf),tab(1),write('Inferences'),tab(2),
225   write('-----'),nl,nl.
226
227 textausgabe(Te) :-
228   name(Te,Text),
229   append("less",Text,C),
230   name(Kommando,C),
231   shell(Kommando,0).
232
233 write_time(X,Y) :-
234   Dauer is Y-X,
```

```
235   convert_time(Dauer,Ye,M,D,H,Mi,S,Milli),
236   Year is Ye-1970,
237   Month is M-1,
238   Day is D-1,
239   Hour is H-1,
240   nl,tab(5),
241   write('-----------------------'),
242   nl,tab(5),
243   write('DasProgrammlief:'),nl,
244  % tab(10),write(Year),tab(1),write('Jahre'),
245  % tab(2),write(Month),tab(1),write('Monate'),
246  % tab(2),write(Day),tab(1),write('Tage'),nl,
247   tab(10),write(Hour),tab(1),write('Stunden'),tab(2),
248   tab(10),write(Mi),tab(1),write('Minuten'),nl,
249   tab(10),write(S),tab(1),write('Sekunden'),tab(2),
250   write(Milli),tab(1),write('Millisekunden'),nl,
251   tab(5),
252   write('-----------------------'),nl,
253   nl.
254
255 /*****************************************/
257 /***                End of file        ***/
259 /*****************************************/
261 %
262 % End of file pred.prolog
```

Datei 3: monitor.prolog

```
1  /*********************************************/
3  /***                Menuemodul               ***/
5  /*********************************************/
7  %
8  %
9  % letzte Aenderung: 19/3/00
10 %
11 % Dateiname: monitor.prolog
12 %
13 % Aufruf erfolgt mit: 'menu/0'
14 %
15 /*********************************************/
17 /***                  menu/0                 ***/
19 /*********************************************/
21 %
22 %
23 % menu/0
24 %
25 menu:-
26   consult('pred.prolog'),
27   nl,
28   write('Programm zur Krisensimulation'),nl,
29   write('----------------------------'),nl,
30   write('Herzlich willkommen und viel Spass'),nl,
31   nl,
32   write('bitte geben Sie die gewuenschte Aktion ein:'),nl,
33   write('----------------------------'),nl,
34   write('1 Eingabe der Parameter'),nl,
35   write('2 Starten der Simulation'),nl,
36   % write('4 Antwort maximal'),nl,
37   write('z zeige Parameter'),nl,
38   write('a zeige Liste der moeglichen Anfangshandlungen'),nl,
39   write('c zeige Charaktere'),nl,
40   write('h zeige die Geschichte'),nl,
41   write('w zeige wie die Perioden enden'),nl,
42   write('e Simulation beenden'),nl,
43   nl,
44   write('Bitte Auswahl eingeben'),nl,
45   nl,
```

```
46   write('Eingabe > '),
47   read_atom(Eingabe),nl,
48   verarbeite(Eingabe),
49   !.% rotes Cut
50
51 /*********************************************/
53 /***                    verarbeite/1          ***/
55 /*********************************************/
57 %
58 % verarbeite(+BUCHSTABE)
59 %
60 % BUCHSTABE ist ein Atom aus Auswahlmenue
61 %
62 % ruft die zum Auswahlmenue gehoerenden Aktionen auf:
63 %      1 Eingabe der Parameter
64 %      2 Starten der Simulation
65 %      z zeige Parameter
66 %      a zeige Liste der moeglichen Anfangshandlungen
67 %      c zeige Charaktere
68 %      h zeige die Geschichte
69 %      w zeige wie die Perioden enden
70 %      e Simulation benden
71 %
72 verarbeite(Atom):-
73   member(Atom,[1,par]),
74   write('Sie waehlten Aenderung der Parameter!'),nl,nl,
75   number_runs,
76   !.% rotes Cut
77
78 verarbeite(Atom):-
79   member(Atom,[2,start]),
80   write('Sie waehlten Start der Simulation!'),nl,nl,
81   programmaufruf,
82   !.% rotes Cut
83
84 verarbeite(Atom):-
85   member(Atom,[z,zeige]),
86   write('Sie wollen die Parameter sehen'),nl,nl,
87   write('Hier ist sie:'),nl,
88   textausgabe('para.prolog'),nl,
89   write('[RETURN] zum Weitermachen druecken ...'),
```

```
90   read_atom(_),nl,
91   menu.% rotes Cut
92
93 verarbeite(Atom):-
94   member(Atom,[h]),
95   write('Sie wollen die Geschichte sehen'),nl,nl,
96   write('Hier ist sie:'),nl,
97   textausgabe('history.prolog'),nl,
98   write('[RETURN] zum Weitermachen druecken ...'),
99   read_atom(_),nl,
100  menu.% rotes Cut
101
102 verarbeite(Atom):-
103  member(Atom,[w]),
104  write('Sie wollen das Ende sehen'),nl,nl,
105  write('Hier ist sie:'),nl,
106  textausgabe('ende.prolog'),nl,
107  write('[RETURN] zum Weitermachen druecken ...'),
108  read_atom(_),nl,
109  menu.% rotes Cut
110
111 verarbeite(Atom):-
112  member(Atom,[c]),
113  write('Sie wollen die Charaktere sehen'),nl,nl,
114  write('Hier sind sie:'),nl,
115  textausgabe('charakter.prolog'),nl,
116  write('[RETURN] zum Weitermachen druecken ...'),
117  read_atom(_),nl,
118  menu.% rotes Cut
119
120 verarbeite(Atom):-
121  member(Atom,[a,anf]),
122  write('Sie wollen die moeglichen Anfangshandlungen sehen'),
123  nl,nl,
124  write('Hier sind sie:'),nl,
125  textausgabe('actions.prolog'),nl,
126  write('[RETURN] zum Weitermachen druecken ...'),
127  read_atom(_),nl,
128  menu.% rotes Cut
129
130 verarbeite(Atom):-
```

```
131   member(Atom,[e,ende,quit,bye,finish,end,exit,halt]),
132   write('Die Krisensimulation wird beendet!'),nl,
133   write('Sind Sie sicher, dass Sie
134           die Simulation beenden wollen?'),nl,
135   get_ja_oder_nein(JaNein1),
136   !,% rotesCut
137   member(JaNein1,[ja,yes,j,y]).
138
139 verarbeite(Atom):-
140   write('Mit'''),
141   write(Atom),
142   write(''' ist leider keine Aktion verbunden!'),nl,
143   nl,
144   write('Wollen sie das Programm beenden?'),nl,
145   get_ja_oder_nein(JaNein1),
146   !,% rotes Cut
147   member(JaNein1,[ja,yes,j,y]).
148
149 /*****************************************/
151 /***                 number_runs/0     ***/
153 /*****************************************/
155 %
156 % number_runs/0
157 %
158 number_runs:-
159   del('para.prolog'),
160   nl,
161   write('-----------------------------'),nl,
162   nl,
163   repeat,
164   write('bitte waehlen Sie die Anzahl
165                     der gewuenschten Runs:'),nl,
166   nl,
167   write('-----------------------------'),nl,
168   write('1 bis 100'),nl,
169   nl,
170   write('Bitte Auswahl eingeben'),nl,
171   nl,
172   write('Eingabe >'),
173   read_integer(Eingabe),nl,
174   verarbeite_runs(Eingabe),
```

```
175   !.% rotes Cut
176
177 /*********************************************/
179 /***                  verarbeite_periods/1          ***/
181 /*********************************************/
183 %
184 % verarbeite_periods(+INTEGER)
185 %
186 % INTEGER ist eine ganze Zahl
187 %
188 % schreibt die ganzen ins Datenfile 'para.prolog':
189 %
190 verarbeite_runs(Integer):-
191   write('Sie waehlten'),tab(1),write(Integer),nl,
192   append('para.prolog'),nl,
193   write('runs('),write(Integer),write(').'),
194   told,
195   number_periods,
196   !.% rotes Cut
197
198 /*********************************************/
200 /***                    number_periods/0           ***/
202 /*********************************************/
204 %
205 % number_periods/0
206 %
207 number_periods:-
208   nl,
209   write('-------------------------'),nl,
210   nl,
211   repeat,
212   write('bitte waehlen Sie die Anzahl
213               der gewuenschten Perioden:'),nl,
214   nl,
215   write('-------------------------'),nl,
216   write('1 bis 5000'),nl,
217   nl,
218   write('Bitte Auswahl eingeben'),nl,
219   nl,
220   write('Eingabe > '),
221   read_integer(Eingabe),nl,
```

```
222   verarbeite_periods(Eingabe),
223   !.% rotes Cut
224
225 /*****************************************/
227 /***                verarbeite_periods/1          ***/
229 /*****************************************/
231 %
232 % verarbeite_periods(+INTEGER)
233 %
234 % INTEGER ist eine ganze Zahl
235 %
236 % schreibt die ganzen ins Datenfile 'para.prolog':
237 %
238 verarbeite_periods(Integer):-
239   Integer < 1000,
240   write('Sie waehlten'),tab(1),write(Integer),nl,
241   append('para.prolog'),nl,
242   write('periods('),write(Integer),write(').'),
243   told,
244   number_actors,
245   !.% rotes Cut
246
247 verarbeite_periods(Integer):-
248   Integer >= 1000,
249   write('Sie waehlten'),tab(1),write(Integer),nl,
250   write('!!! ACHTUNG !!!'),nl,
251   write('Es muss mit einer laengeren
252                Verarbeitungszeit gerechnet werden!!'),nl,
253   write('Sind Sie sicher?'),nl,
254   get_ja_oder_nein(JaNein1),
255   !,% rotes Cut
256   member(JaNein1,[ja,yes,j,y]),
257   append('para.prolog'),nl,
258   write('periods('),write(Integer),write(').'),
259   told,
260   number_actors,
261   !.% rotes Cut
262
263 /*****************************************/
265 /***                number_actors/0          ***/
267 /*****************************************/
```

```
269 %
270 % number_actors/0
271 %
272 number_actors :-
273   nl,
274   write('-----------------------------'),nl,
275   nl,
276   repeat,
277   write('bitte waehlen Sie die Anzahl
278                 der gewuenschten Akteure:'),nl,
279   nl,
280   write('-----------------------------'),nl,
281   write('1 bis 100'),nl,
282   nl,
283   write('Bitte Auswahl eingeben'),nl,
284   nl,
285   write('Eingabe > '),
286   read_integer(Eingabe),nl,
287   verarbeite_actors(Eingabe),
288   !.% rotes Cut
289
290 /*****************************************/
292 /***                verarbeite_actors/1          ***/
294 /*****************************************/
296 %
297 % verarbeite_actors(+INTEGER)
298 %
299 % INTEGER ist eine ganze Zahl
300 %
301 % schreibt die ganzen ins Datenfile 'para.prolog':
302 %
303 %
304 verarbeite_actors(Integer) :-
305   write('Sie waehlten'),tab(1),write(Integer),nl,
306   append('para.prolog'),nl,
307   write('actors('),write(Integer),write(').'),
308   told,
309   number_charakters,
310   !.% rotes Cut
311
312 /*****************************************/
```

```
/***                    number_charakters/0          ***/
/*****************************************************/
%
% number_charakters/0
%
number_charakters :-
  nl,
  write('-----------------------------'),nl,
  nl,
  repeat,
  write('bitte waehlen Sie ob die Charaktere ueber
         die Runs beibehalten werden sollen'),nl,
  nl,
  write('-----------------------------'),nl,
  write('"yes" oder"no"'),nl,
  nl,
  write('Bitte Auswahl eingeben'),nl,
  nl,
  write('Eingabe > '),tab(1),nl,
  get_ja_oder_nein(JaNein2),nl,
  verarbeite_charakters(JaNein2),
  !.% rotes Cut

/*****************************************************/
/***                  verarbeite_charakters/1        ***/
/*****************************************************/
%
% verarbeite_charakters(+Atom)
%
% INTEGER ist eine ganze Zahl
%
% schreibt die ganzen ins Datenfile 'para.prolog':
%
verarbeite_charakters(JaNein2) :-
  member(JaNein2,[ja,yes,j,y]),
  write('Sie waehlten'),tab(1),write(yes),nl,
  append('para.prolog'),nl,
  write('use_old_data('),write(yes),write(').'),
  told,
  programmaufruf,
  !.% rotes Cut
```

```
360
361 verarbeite_charakters(JaNein2):-
362   member(JaNein2,[no,nein,n]),
363   write('Sie waehlten'),tab(1),write(no),nl,
364   append('para.prolog'),nl,
365   write('use_old_data('),write(no),write(').'),
366   told,
367   programmaufruf,
368   !.% rotes Cut
369
370 /*********************************************/
372 /***                    read_atom/1          ***/
374 /*********************************************/
376 %
377 % read_atom(-ATOM)
378 %
379 % ATOM ist ein Atom.
380 %
381 % Liest jede Zeichenkette, die mit [RETURN] eingegeben wird,
382 % als Atom ein.
383 % Die Eingaben koennen auch mit Grossbuchstaben beginnen
384 % oder Sonderzeichen enthalten.
385 %
386 read_atom(Atom):-
387   get0(C),
388   complete_atom(C,List),
389   name(Atom,List),
390   !.% gruenes Cut
391
392 /*********************************************/
394 /***                  read_integer/1         ***/
396 /*********************************************/
398 %
399 % read_integer(-INTEGER)
400 %
401 % INTEGER ist eine ganze Zahl.
402 %
403 % Liest jedes Zeichen, das mit [RETURN] eingegeben wird,
404 % als Integer ein.
405 %
406 %
```

```
407 %
408 read_integer(Integer):-
409   get0(C),
410   complete_integer(C,List),
411   name(Integer,List),
412   !.% gruenes Cut
413
414 /*****************************************/
416 /***                 complete_atom/2     ***/
418 /*****************************************/
420 %
421 % complete_atom(+ASCII,-ASCII_LISTE)
422 %
423 % ASCII eine positive ganze Zahl 0...255 (ASCII-Code).
424 % ASCII_LISTE  eine Liste mit ASCII-Codes.
425 %
426 % liest solange Zeichen ein bis [RETURN] eingegeben wird.
427 %
428 complete_atom(10,[]):-/*10=Return*/
429   !.%  rotes Cut
430
431 complete_atom(C1,[C1|Rest]):-
432   get0(C2),
433   complete_atom(C2,Rest).
434
435 /*****************************************/
437 /***                 complete_integer/2  ***/
439 /*****************************************/
441 %
442 % complete_integer(+INTEGER,-INTEGER_LISTE)
443 %
444 % INTEGER eine positive ganze Zahl.
445 % INTEGER_LISTE  eine Liste mit ganzen Zahlen.
446 %
447 % liest solange Zeichen ein bis [RETURN] eingegeben wird.
448 %
449 complete_integer(10,[]):-/*10=Return*/
450   !.% rotes Cut
451
452 complete_integer(C1,[C1|Rest]):-
453   get0(C2),
```

```
454   complete_integer(C2,Rest).
455
456 /*****************************************/
458 /***                 get_ja_oder_nein/1      ***/
460 /*****************************************/
462 %
463 % get_ja_oder_nein(-JA-NEIN)
464 %
465 %
466 %
467 % fragt solange bis 'ja', 'nein', 'j', 'n', 'yes', 'no', oder
468 % 'y' eingegeben wird
469 %
470 get_ja_oder_nein(JaNein):-
471   repeat,
472   write('Eingabe(jodern) >'),
473   read_atom(JaNein),nl,
474   member(JaNein,[ja,nein,j,n,yes,no,y]),
475   !.% rotes Cut
476
477 /*****************************************/
479 /***                    get_number/1         ***/
481 /*****************************************/
483 %
484 % get_number(-INTEGER)
485 %
486 % INTEGER ist eine ganze Zahl
487 %
488 % fragt solange bis eine ganze Zahl eingegeben
489 % wird
490 %
491 get_number(Integer):-
492   repeat,
493   write('Eingabe < (1-10) >'),
494   read_integer(Integer),nl,
495   integer(Integer),
496   !.% rotes Cut
497
498 /*****************************************/
500 /***                 pp_handlung/0           ***/
502 /*****************************************/
```

```
504 % pp_handlung/0
505 %
506 % Gibt das Wissen der Faktenbasis als Liste aus.
507 %
508 pp_handlung :-
509   lade_wissen,
510   write('handeln(Familie,Gruppe,Handlung,Staerke,Maxgrad,
511               Kategorie,Ausrichtung)'),nl,
512   (handeln(Familie,Gruppe,Handlung,Staerke,Maxgrad,
513               Kategorie,Ausrichtung)),
514   write(handeln(Familie,Gruppe,Handlung,Staerke,Maxgrad,
515               Kategorie,Ausrichtung)),
516   nl,
517   fail
518 ;
519   retractall(handeln),
520   true.
521
522 /*******************************************/
524 /***                pp_anfang/0          ***/
526 /*******************************************/
528 % pp_anfang/0
529 %
530 % Gibt die moeglichen Anfangshandlungen als Liste aus.
531 %
532 pp_anfang :-
533   lade_anfang,
534   write('handeln(Familie,Gruppe,Handlung,Staerke,Maxgrad,
535               Kategorie,Ausrichtung)'),nl,
536   (handeln(Familie,Gruppe,Handlung,Staerke,Maxgrad,
537               Kategorie,Ausrichtung)),
538   write(handeln(Familie,Gruppe,Handlung,Staerke,Maxgrad,
539               Kategorie,Ausrichtung)),
540   nl,
541   fail
542 ;
543   retractall(handeln),
544   true.
545
546 /*******************************************/
548 /***                lade_wissen/0        ***/
```

```
/*****************************************/
% lade_wissen/0
%
% Laedt die Datei 'faktenbasis.prolog', falls sie
% existiert. Falls nicht, wird die Datei
% 'faktenbasis.prolog ~' geladen. Falls diese auch nicht
% existiert, wird die Daten-basis nur initialisiert.
%
lade_wissen :-
  exists_file('faktenbasis.prolog'),
  write('Die Faktenbasisbasis aus der Datei
            "faktenbasis.prolog" wird geladen ...'),
  consult('faktenbasis.prolog'),
  write(fertig),nl,
  !.% rotes Cut

lade_wissen :-
  exists_file('faktenbasis.prolog ~'),
  write('Die Wissensbasis aus "faktenbasis.bak" wird geladen ...'),
  consult('faktenbasis.prolog ~'),
  write(fertig),nl,
  !.% rotesCut

lade_wissen :-% nur Initialisierung
  assertz(handeln(familie,gruppe,handlung,staerke,maxgrad,
            kategorie,ausrichtung)).

/*****************************************/
/***                    lade_anfang/0              ***/
/*****************************************/
% lade_anfang/0
%
% Laedt die Datei 'anfangsliste.prolog', falls sie
% existiert. Falls nicht, wird die Datei
% 'anfangsliste.prolog ~' geladen.
% Falls diese auch nicht existiert, wird die
%   Datenbasis nur initialisiert.
%
lade_anfang :-
  exists_file('anfangsliste.prolog'),
  write('Die Liste der moeglichen Starthandlungen aus
```

```
595           der Datei "anfangsliste.prolog" wird geladen ...'),
596   consult('anfangsliste.prolog'),
597   write(fertig),nl,
598   !.% rotes Cut
599
600 lade_anfang :-
601   exists_file('anfangsliste.prolog ~'),
602   write('Die Liste der moeglichen Starthandlungen aus
603           der Datei "anfangsliste.prolog" wird geladen ...'),
604   consult('anfangsliste.prolog ~'),
605   write(fertig),nl,
606   !.% rotes Cut
607
608 lade_anfang :-% nur Initialisierung
609   assertz(handeln(familie,gruppe,handlung,staerke,maxgrad,
610           kategorie,ausrichtung)).
611
612 /*********************************************/
614 /***                sammle/1               ***/
616 /*********************************************/
618 %
619 % sammle/1
620 %
621 %
622 % sammle(L) :-
623 % retract(stack(X)),!,% entferne naechste Loesung
624 % (X == ende,!,L = []; %   Ende der Loesung
625 % L = [X | Rest],
626 % sammle(Rest)).% ansonsten sammle Rest
627 %
628 sammle(L) :-
629   retract(stack(X)),!,
630   (X == ende,!,L = []
631 ;
632   L = [X | Rest],
633   sammle(Rest)).
634
635 /*********************************************/
637 /***              lese_satz/1              ***/
639 /*********************************************/
641 %
```

```
642 % lese_satz(?Liste)
643 %
644 % Die Prozedur liest einen Satz ein
645 %
646 lese_satz(Liste):-
647   get0(Zeichen),
648   lese_rest(Zeichen,Liste).
649
650 lese_rest(46,[]).
651
652 lese_rest(32,Liste):-
653   lese_satz(Liste).
654
655 lese_rest(91,Liste):-
656   lese_satz(Liste).
657
658 lese_rest(93,Liste):-
659   lese_satz(Liste).
660
661 lese_rest(Buchstabe,[Wort|Liste]):-
662   lese_buchstabe(Buchstabe,Buchstaben,Naechstes_zeichen),
663   name(Wort,Buchstaben),
664   lese_rest(Naechstes_zeichen,Liste).
665
666 lese_buchstabe(46,[],46):-!.
667
668 lese_buchstabe(32,[],32):-!.
669
670 lese_buchstabe(B,[B|Buchstaben],Naechstes_zeichen):-
671   get0(Zeichen),
672   lese_buchstabe(Zeichen,Buchstaben,Naechstes_zeichen).
673
674 /*****************************************/
676 /***                islist/1           ***/
678 /*****************************************/
680 %
681 % islist(?Liste)
682 %
683 % Erkennung bzw. Generierung von Listen
684 %
685 islist([]).
```

```
686
687 islist([_|Tail]):-
688   islist(Tail).
689
690 /*****************************************/
692 /***                programmaufruf/0       ***/
694 /*****************************************/
696 %
697 % Hilfspraedikat fuer Ausnahmebehandlung
698 %
699 programmaufruf:-
700   consult('prog.prolog'),
701   start,
702   menu.
703
704 /*****************************************/
706 /***                End of file            ***/
708 /*****************************************/
710 %
711 % End of file monitor.prolog
```

Datei 4: feasable.prolog.

```
1  /*******************************************/
3  /***                feasable                ***/
5  /*******************************************/
7  %
8  % letzte Aenderung: 19/3/00
9  % Dateiname: feasable.prolog
10 % Diese Datei kontrolliert die Durchfuehrbarkeit
11 % von Handlungen.
12 % feasable/1
13 % feasable(action(Run,Periode,Akteur,Ausrichtung,Handlung,
14 %    Staerkegrad,Maximalstaerke,Nummer).
15 %
16 feasable(action(R,T,AS,Direction,Handlungsname,Neustaerke,
17                                              Maxstaerke,Nr)):-
18   true,!.
```

Datei 5: flags.prolog

```
/*******************************************/
/***                 Zaehler             ***/
/*******************************************/
%
%
% letzte Aenderung: 19/3/00
%
% Dateiname: flags.prolog
%
% Diese Datei verwaltet die Zaehler
%
make_flags:-
  action_flags,
  end_flags.

action_flags:-
  flag(gesamt,_,0),
  flag(lives,_,0),
  flag(periodnow,_,0),
  findall(Nummer,act(_,_,_,Nummer),Anzahl),
  length(Anzahl,Laenge),
  (between(1,Laenge,X),
  nth1(X,Anzahl,Y),
  act(H,_,_,Y),
  %(between(1,3,Z),
  nth1(1,[positiv,negativ,unterlassung],Dir),
  flagname_run(H,Dir,F),
  flag(F,_,0),
  flagname_gesamt(H,Dir,H1),
  flag(H1,_,0),
  fail
;true,!).

% Alle Zaehler werden auf Null gesetzt

reset_flags:-
  findall(Nummer,act(_,_,_,Nummer),Anzahl),
  length(Anzahl,Laenge),
  (between(1,Laenge,X),
```

```
43    nth1(X,Anzahl,Y),
44    act(H,_,_,Y),
45    %(between(1,3,Z),
46    nth1(1,[positiv,negativ,unterlassung],Dir),
47    flagname_run(H,Dir,F),
48    flag(F,_,0),
49    %fail; true),
50    fail
51  ; true,!).
52
53  % Die Zaehler werden in die Datei 'Ergebnis' geschrieben
54
55  write_flagsrun :-
56    flag(lives,Z,Z),
57    flag(periodnow,U,U),
58    run(R),
59    charakter(1,Liste1,Staerke1,Fanatiker1),
60    C1lis = [1,Liste1,Staerke1,Fanatiker1],
61    charakter(2,Liste2,Staerke2,Fanatiker2),
62    C2lis = [2,Liste2,Staerke2,Fanatiker2],
63    periods(T),
64    findall(Nummer,act(_,_,_,Nummer),Anzahl),
65    length(Anzahl,Laenge),
66    asserta(hlis([])),
67    (between(1,Laenge,X),
68    hlis(Old),
69    nth1(X,Anzahl,Y),
70    act(H,_,_,Y),
71    nth1(1,[positiv,negativ,unterlassung],Dir),
72    flagname_run(H,Dir,F),
73    flag(F,E,E),
74    mittel(E,U,M),
75    append(Old,[M],H1lis),
76    (retract(hlis(_)),fail
77  ; true),
78    asserta(hlis(H1lis)),
79    fail
80  ; true,!),
81    hlis(New),
82    append('Ergebnis'),
83    write('fact('),write(C1lis),write(','),
```

```
84    write(C2lis),write(','),
85    write(New),
86    write(').'),
87    nl,
88    told.
89
90  % Die Zaehler werden in die Datei 'tabellegesamt' geschrieben
91
92  write_flagsgesamt :-
93    flag(lives,Z,Z),
94    findall(Nummer,act(_,_,_,Nummer),Anzahl),
95    length(Anzahl,Laenge),
96    runs(RX),
97    flag(gesamt,Wer,Wer),
98    append('tabellegesamt'),
99    write('Anzahl der ausgefuehrten Handlungen :'),tab(1),
100   write(Wer),
101   nl,
102   write_endflags,
103   told,
104   (between(1,Laenge,X),
105   nth1(X,Anzahl,Y),
106   act(H,_,_,Y),
107   % (between(1,3,Z1),
108   nth1(1,[positiv,negativ,unterlassung],Dir),
109   flagname_gesamt(H,Dir,H1),
110   flag(H1,E1,E1),
111   mittel(E1,Z,M),
112   mittel(M,RX,M1),
113   append('tabellegesamt'),
114   write(H1),write(','),tab(1),write(E1),write(','),tab(1),write(M),
115   write(','),tab(1),write(M1),
116   nl,
117   told,
118   % fail; true),
119   fail
120 ; true).
121
122 /*****************************************/
124 /***                Hilfspraedikate        ***/
126 /*****************************************/
```

```
128 %
129 mittel(E,U,M):-
130   U > 0,
131   M is E/U,
132   !.
133
134 flagname_run(H,Dir,F):-
135   concat(H,'_',Hn),
136   concat(Hn,Dir,F).
137
138 flagname_gesamt(H,Dir,H1):-
139   concat(H,'_',Hn),
140   concat(Hn,0,H0),
141   concat(H0,Dir,H1).
142
143 rule_flags:-
144   findall(Art,rule(Art),Rules),
145   asserta(rule_list(Rules)),
146   length(Rules,Laenge),
147   (between(1,Laenge,X),
148   nth1(X,Rules,Name),
149   concat(Name,0,N0),
150   flag(N0,_,0),
151   concat(Name,1,N1),
152   flag(N1,_,0),fail
153 ;true,!).
154
155 count_flag(Handlungsname,Direktion):-
156   flagname_run(Handlungsname,Direktion,F),
157   flag(F,Z,Z+1),
158   flagname_gesamt(Handlungsname,Direktion,H1),
159   flag(H1,G,G+1),
160   flag(gesamt,W,W+1),
161   flag(periodnow,U,U+1).
162
163 delete_tabs:-
164   concat('tabelle','*',Tab),
165   name(Tab,T),
166   append("rm",T,C),
167   name(Kommando,C),
168   shell(Kommando)
```

```
169 ; true.
170
171 end_flags :-
172   flag(ende_aufgeben,_,0),
173   flag(ende_grossangriff,_,0),
174   flag(ende_periodsmax,_,0).
175
176 write_endflags :-
177   flag(ende_aufgeben,S,S),
178   flag(ende_grossangriff,G,G),
179   flag(ende_periodsmax,M,M),
180   flag(grossangriff_0positiv,P,P),
181   flag(aufgeben_0positiv,A,A),
182   flag(lives,L,L),
183   runs(R),
184   Sum is L*R,
185   Auf is S+A,
186   Gr is G+P,
187   write('AnzahlderRuns:'),
188   write(Sum),tab(1),
189   write('aufgebenR:'),tab(1),write(S),tab(2),
190   write('grossangriffR:'),tab(1),
191   write(G),tab(2),write('Max.Anz.Perioden:'),tab(1),write(M),nl,
192   write('aufgebenges:'),tab(1),write(Auf),tab(2),
193   write('grossangriffges:'),tab(1),write(Gr),nl.
194
195 /*****************************************/
197 /***                End of file        ***/
199 /*****************************************/
201 %
202 % End of file flags.prolog
```

Datei 6: Beispiel para.prolog.

```
/*********************************************/
/***                    Parameterdatei       ***/
/*********************************************/
%
% letzte Aenderung: 20/3/00
% Dateiname: para.prolog
% Parameterdatei
%
runs(20).
periods(100).
actors(2).
use_old_data(no).

/*********************************************/
/***                    End of file          ***/
/*********************************************/
% End of file para.prolog
```

Datei 7: actions.prolog

```
/*********************************************/
/***                Handlungen                ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: actions.prolog
% Diese Datei ist eine Faktenbasis der Handlungen:
% act/4
% act(Handlung,Staerkegrad1,Maximalstaerke,Nummer).
%
act(angriffsdrohung,1,5,1).
act(beschuldigung,1,3,2).
act(vergeltung,1,5,3).
act(sanktionen,1,3,4).
act(miskredit,1,3,5).
act(erschweren,1,2,6).
act(handel,1,3,7).
act(unterdrueckung,1,3,8).
act(verunglimpfung,1,3,9).
act(embargo,1,5,10).
act(blockade,1,4,11).
act(zurueckhalten,1,3,12).
act(manoever,1,4,13).
act(mobilmachung,1,3,14).
act(truppenverlegung,1,3,15).
act(grenzueberschreitung,1,3,16).
act(luftraum,1,4,17).
act(schiffeversenken,1,3,18).
act(bombardieren,1,5,19).
act(grossangriff,1,1,20).
act(waffenproduktion,1,3,21).
act(konstruktion,1,4,22).
act(anlagenbau,1,3,23).
act(legitimation,1,2,24).
act(aufruf,1,3,25).
act(unterminieren,1,4,26).
act(sabotage,1,3,27).
act(terror,1,4,28).
act(demonstration,1,3,29).
act(streik,1,3,30).
act(skip,1,1,31).
act(aufgeben,1,1,32).
```

Datei 8: Beispiel charakter.prolog, bzw. cha.prolog

```
/*********************************************/
/***                Charakter.prolog             ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: charakter.prolog
% DieseDateidefiniertdieverwendetenCharaktere
% actor_list/1
% charakter/4
%
actor_list([1,2]).

% ------------- Akteur 1 ------------
% Regeln:[linear,titfortat,tfts,desk,lingr,deskgr,lin2,desk2,maxesk]
% Staerke:7 Fanatiker:yes
charakter(1,[1.84502,15.1292,18.2657,10.1476,6.64207,12.5461,15.8672,5.71956,13.8376],7,yes).
% ------------- Akteur 2 ------------
% Regeln:[linear,titfortat,tfts,desk,lingr,deskgr,lin2,desk2,maxesk]
% Staerke:38 Fanatiker:no
charakter(2,[6.17849,5.72082,11.4416,14.1876,20.3661,4.57666,16.476,1.60183,19.4508],38,no).

/*********************************************/
/***                End of file                ***/
/*********************************************/
% End of file charakter.prolog
```

Datei 9: charakters.prolog

```
/*********************************************/
/***                Charaktere                ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: charakters.prolog
% Diese Datei ist eine Faktenbasis der Charaktere:
% charakter_arts/2
% charakter_arts(Name,Liste).
%
charakter_arts(agressiv,[30,2,2,2,30,2,30,1.95,0.05]).
charakter_arts(ausgeglichen,[12,13,13,13,12,12,12,12.99,0.01]).
charakter_arts(deeskalierend,[1,1,5,30,1,30,1,30.99,0.01]).
charakter_arts(tft,[4,40,40,3,3,3,3,3.99,0.01]).
charakter_arts(tftdesk,[1,19,20,19,1,19,1,19.99,0.01]).
charakter_arts(tftesk,[19,20,20,1,19,1,19,0.99,0.01]).
charakter_arts(absolutdeeskalierend,[0,0,0,35,0,35,0,29.99,0.01]).
```

Datei 10: glcha.prolog.

```
/*********************************************/
/***                 Charaktere                ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: glcha.prolog
% Diese Datei ist eine Faktenbasis der Charaktere:
% charakter_arts/2
% charakter_arts(Name,Liste).
% gleiche Charaktere fuer den Kontrolllauf.
%
charakter_arts(agressiv,[12,13,13,13,12,12,12,12.99,0.01]).
charakter_arts(ausgeglichen,[12,13,13,13,12,12,12,12.99,0.01]).
charakter_arts(deeskalierend,[12,13,13,13,12,12,12,12.99,0.01]).
charakter_arts(tft,[12,13,13,13,12,12,12,12.99,0.01]).
charakter_arts(tftdesk,[12,13,13,13,12,12,12,12.99,0.01]).
charakter_arts(tftesk,[12,13,13,13,12,12,12,12.99,0.01]).
charakter_arts(absolutdeeskalierend,[12,13,13,13,12,12,12,12.99,0.01]).
```

Datei 11: 92cha.prolog.

```
/*********************************************/
/***                 Charaktere                ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: 92cha.prolog
% Diese Datei ist eine Faktenbasis der Charaktere:
% charakter_arts/2
% charakter_arts(Name,Liste).
% gleiche Charaktere fuer den Kontrolllauf.
%
charakter_arts(linear,[92,1,1,1,1,1,1,1.99,0.01]).
charakter_arts(lingr,[1,92,1,1,1,1,1,1.99,0.01]).
charakter_arts(lin2,[1,1,92,1,1,1,1,1.99,0.01]).
charakter_arts(desk,[1,1,1,92,1,1,1,1.99,0.01]).
charakter_arts(deskgr,[1,1,1,1,92,1,1,1.99,0.01]).
charakter_arts(desk2,[1,1,1,1,1,92,1,1.99,0.01]).
charakter_arts(titfortat,[1,1,1,1,1,1,92,1.99,0.01]).
```

Datei 12: Beispiel rules.prolog.

```
/*********************************************/
/***                    Regelliste          ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: rules.prolog
% Diese Datei listest alle verwendeten Regeln auf.
%
% rule/1
%
rule(linear).
rule(lingr).
rule(lin2).
rule(desk).
rule(deskgr).
rule(desk2).
rule(titfortat).
rule(tfts).
rule(maxesk).

/*********************************************/
/***                  End of file           ***/
/*********************************************/
% End of file rules.prolog
```

Datei 13: linear

```
/*********************************************/
/***                Regel:linear             ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: linear
% Diese Datei beschreibt die Regel: linear
% linear/5
% linear(Runs,Perioden,Akteur,Staerke,Nummer).
%
linear(R,T,AS,Staerke,Nr):-
  Vorher is T-1,
  action(R,Vorher,_,Direktion,Handlungsname,Staerke,
                  Maxstaerke,Nr),
  Staerke<Maxstaerke,
  NS is Staerke+1,
  Handlung=Handlungsname,
  direktion_ini,
  direktion(D),
  asserta(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),
  (retract(choosed_actionart(act(_,1,_,_))); true),
  asserta(choosed_actionart(act(Handlung,1,Maxstaerke,Nr))),
  feasable(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),!.

linear(R,T,AS,Staerke,Nr):-
  Vorher is T-1,
  action(R,Vorher,_,Direktion,Handlungsname,Staerke,
                  Maxstaerke,Nr),
  Staerke=Maxstaerke,
  esk(Handlungsname,Handlung),
  act(Handlung,1,X,Num),
  direktion_ini,
  direktion(D),
  asserta(action(R,T,AS,D,Handlung,1,X,Num)),
  (retract(choosed_actionart(act(_,1,_,_))); true),
  asserta(choosed_actionart(act(Handlung,1,Max,Num))),
  feasable(action(R,T,AS,D,Handlung,1,Max,Num)),!.

esk(miskredit,erschweren).
```

```
45 esk(miskredit,miskredit).
46 esk(erschweren,handel).
47 esk(handel,sanktionen).
48 esk(sanktionen,zurueckhalten).
49 esk(zurueckhalten,manoever).
50 esk(manoever,verunglimpfung).
51 esk(verunglimpfung,aufruf).
52 esk(aufruf,demonstration).
53 esk(demonstration,unterminieren).
54 esk(unterminieren,streik).
55 esk(streik,mobilmachung).
56 esk(mobilmachung,truppenverlegung).
57 esk(truppenverlegung,embargo).
58 esk(embargo,waffenproduktion).
59 esk(waffenproduktion,konstruktion).
60 esk(konstruktion,anlagenbau).
61 esk(anlagenbau,beschuldigung).
62 esk(beschuldigung,angriffsdrohung).
63 esk(angriffsdrohung,vergeltung).
64 esk(vergeltung,legitimation).
65 esk(legitimation,sabotage).
66 esk(sabotage,unterdrueckung).
67 esk(unterdrueckung,terror).
68 esk(terror,blockade).
69 esk(blockade,grenzueberschreitung).
70 esk(grenzueberschreitung,luftraum).
71 esk(luftraum,schiffeversenken).
72 esk(schiffeversenken,bombardieren).
73 esk(bombardieren,grossangriff).
74 esk(skip,miskredit).
75 esk(skip,skip).
76 esk(grossangriff,grossangriff).
77 esk(aufgeben,aufgeben).
78 %
79 /*****************************************/
81 /***                End of file          ***/
83 /*****************************************/
```

Datei 14: lingr

```
/*********************************************/
/***                 Regel:lingr                ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: lingr
% Diese Datei beschreibt die Regel: lingr
% lingr/5
% lingr(Runs,Perioden,Akteur,Staerke,Nummer).
%
lingr(R,T,AS,Staerke,Nr):-
  Vorher is T-1,
  action(R,Vorher,_,Direktion,Handlungsname,Staerke,
                  Maxstaerke,Nr),
  Staerke<Maxstaerke,
  NS is Staerke+1,
  Handlung=Handlungsname,
  direktion_ini,
  direktion(D),
  asserta(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),
  (retract(choosed_actionart(act(_,1,_,_))); true),
  asserta(choosed_actionart(act(Handlung,1,Maxstaerke,Nr))),
  feasable(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),!.

lingr(R,T,AS,Staerke,Nr):-
  Vorher is T-1,
  action(R,Vorher,_,Direktion,Handlungsname,Staerke,
                  Maxstaerke,Nr),
  Staerke=Maxstaerke,
  gr(Handlungsname,Handlung),
  act(Handlung,1,X,Num),
  direktion_ini,
  direktion(D),
  asserta(action(R,T,AS,D,Handlung,1,X,Num)),
  (retract(choosed_actionart(act(_,1,_,_))); true),
  asserta(choosed_actionart(act(Handlung,1,Max,Num))),
  feasable(action(R,T,AS,D,Handlung,1,Max,Num)),!.

% Gruppe 1: verbal
```

```
45 gr(angriffsdrohung,beschuldigung).
46 gr(angriffsdrohung,angriffsdrohung).
47 gr(beschuldigung,vergeltung).
48 gr(vergeltung,sanktionen).
49 gr(sanktionen,sanktionen).
50 % Gruppe 2: physikalisch--politisch
51 gr(miskredit,erschweren).
52 gr(miskredit,miskredit).
53 gr(erschweren,handel).
54 gr(handel,unterdrueckung).
55 gr(unterdrueckung,verunglimpfung).
56 gr(verunglimpfung,verunglimpfung).
57 % Gruppe 3: physikalisch--oekonomisch
58 gr(embargo,blockade).
59 gr(embargo,embargo).
60 gr(blockade,zurueckhalten).
61 gr(zurueckhalten,zurueckhalten).
62 % Gruppe 4: physikalisch--gewaltlos militaerisch
63 gr(manoever,mobilmachung).
64 gr(manoever,manoever).
65 gr(mobilmachung,truppenverlegung).
66 gr(truppenverlegung,truppenverlegung).
67 % Gruppe 5: physikalisch--gewaltsam militaerisch
68 gr(grenzueberschreitung,luftraum).
69 gr(grenzueberschreitung,grenzueberschreitung).
70 gr(luftraum,schiffeversenken).
71 gr(schiffeversenken,bombardieren).
72 gr(bombardieren,grossangriff).
73 gr(grossangriff,grossangriff).
74 % Gruppe 6: gewaltfreie militaerische Aktion
75 gr(waffenproduktion,konstruktion).
76 gr(waffenproduktion,waffenproduktion).
77 gr(konstruktion,anlagenbau).
78 gr(anlagenbau,anlagenbau).
79 % Gruppe 7: politische Aktion
80 gr(legitimation,legitimation).
81 % Gruppe 8: verbale interne Aktion
82 gr(aufruf,unterminieren).
83 gr(aufruf,aufruf).
84 gr(unterminieren,unterminieren).
85 % Gruppe 9: physikalische Veraenderung fuer die
```

```
86 %                    amtsinhabende Elite
87 gr(sabotage,terror).
88 gr(sabotage,sabotage).
89 gr(terror,demonstration).
90 gr(demonstration,streik).
91 gr(streik,streik).
92 % Gruppe 10: skip und aufgeben
93 gr(skip,miskredit).
94 gr(skip,skip).
95 gr(aufgeben,aufgeben).
96 %
97 /*****************************************/
99 /***                End of file          ***/
101 /*****************************************/
```

Datei 15: lin2.

```
/*********************************************/
/***                 Regel:lin2                ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: lin2
% Diese Datei beschreibt die Regel: lin2
% lin2/5
% lin2(Runs,Perioden,Akteur,Staerke,Nummer).
%
lin2(R,T,AS,Staerke,Nr):-
  Vorher is T-1,
  action(R,Vorher,_,Direktion,Handlungsname,Staerke,
              Maxstaerke,Nr),
  esk(Handlungsname,H1),esk(H1,Handlung),
  act(Handlung,1,X,Num),
  direktion_ini,
  direktion(D),
  asserta(action(R,T,AS,D,Handlung,1,X,Num)),
  (retract(choosed_actionart(act(_,1,_,_))); true),
  asserta(choosed_actionart(act(Handlung,1,Max,Num))),
  feasable(action(R,T,AS,D,Handlung,1,Max,Num)),!.
```

Datei 16: desk

```
1  /*****************************************/
3  /***                Regel:desk           ***/
5  /*****************************************/
7  %
8  % letzte Aenderung: 19/3/00
9  % Dateiname: desk
10 % Diese Datei beschreibt die Regel: desk
11 % desk/5
12 % desk(Runs,Perioden,Akteur,Staerke,Nummer).
13 %
14 desk(R,T,AS,Staerke,Nr):-
15   Vorher is T-1,
16   action(R,Vorher,_,Direktion,Handlungsname,Staerke,
17               Maxstaerke,Nr),
18   Staerke>1,
19   NS is Staerke-1,
20   Handlung=Handlungsname,
21   direktion_ini,
22   direktion(D),
23   asserta(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),
24   (retract(choosed_actionart(act(_,1,_,_))); true),
25   asserta(choosed_actionart(act(Handlung,1,Maxstaerke,Nr))),
26   feasable(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),!.
27
29 desk(R,T,AS,Staerke,Nr):-
30   Vorher is T-1,
31   action(R,Vorher,_,Direktion,Handlungsname,Staerke,
32               Maxstaerke,Nr),
33   Staerke=1,
34   esk(Handlung,Handlungsname),
35   act(Handlung,1,X,Num),
36   direktion_ini,
37   direktion(D),
38   asserta(action(R,T,AS,D,Handlung,1,X,Num)),
39   (retract(choosed_actionart(act(_,1,_,_))); true),
40   asserta(choosed_actionart(act(Handlung,1,Max,Num))),
41   feasable(action(R,T,AS,D,Handlung,1,Max,Num)),!.
```

Datei 17: deskgr

```
1  /*******************************************/
3  /***                 Regel:deskgr          ***/
5  /*******************************************/
7  %
8  % letzte Aenderung: 19/3/00
9  % Dateiname: deskgr
10 % Diese Datei beschreibt die Regel: deskgr
11 % deskgr/5
12 % deskgr(Runs,Perioden,Akteur,Staerke,Nummer).
13 %
14 deskgr(R,T,AS,Staerke,Nr):-
15   Vorher is T-1,
16   action(R,Vorher,_,Direktion,Handlungsname,Staerke,
17               Maxstaerke,Nr),
18   Staerke>1,
19   NS is Staerke-1,
20   Handlung=Handlungsname,
21   direktion_ini,
22   direktion(D),
23   asserta(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),
24   (retract(choosed_actionart(act(_,1,_,_))); true),
25   asserta(choosed_actionart(act(Handlung,1,Maxstaerke,Nr))),
26   feasable(action(R,T,AS,D,Handlung,NS,Maxstaerke,Nr)),!.
27
29 deskgr(R,T,AS,Staerke,Nr):-
30   Vorher is T-1,
31   action(R,Vorher,_,Direktion,Handlungsname,Staerke,
32               Maxstaerke,Nr),
33   Staerke=1,
34   gr(Handlung,Handlungsname),
35   act(Handlung,1,X,Num),
36   direktion_ini,
37   direktion(D),
38   asserta(action(R,T,AS,D,Handlung,1,X,Num)),
39   (retract(choosed_actionart(act(_,1,_,_))); true),
40   asserta(choosed_actionart(act(Handlung,1,Max,Num))),
41   feasable(action(R,T,AS,D,Handlung,1,Max,Num)),!.
```

Datei 18: desk2.

```
1  /*********************************************/
3  /***                    Regel:desk2                    ***/
5  /*********************************************/
7  %
8  % letzte Aenderung: 19/3/00
9  % Dateiname: desk2
10 % Diese Datei beschreibt die Regel: desk2
11 % desk2/5
12 % desk2(Runs,Perioden,Akteur,Staerke,Nummer).
13 %
14 desk2(R,T,AS,Staerke,Nr):-
15   Vorher is T-1,
16   action(R,Vorher,_,Direktion,Handlungsname,Staerke,
17               Maxstaerke,Nr),
18   esk(H2,Handlungsname),esk(Handlung,H2),
19   act(Handlung,1,X,Num),
20   direktion_ini,
21   direktion(D),
22   asserta(action(R,T,AS,D,Handlung,1,X,Num)),
23   (retract(choosed_actionart(act(_,1,_,_))); true),
24   asserta(choosed_actionart(act(Handlung,1,Max,Num))),
25   feasable(action(R,T,AS,D,Handlung,Staerke,Max,Num)),!.
```

Datei 19: titfortat.

```
/*********************************************/
/***                    Regel:titfortat      ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: titfortat
% Diese Datei beschreibt die Regel: titfortat
% titfortat/5
% titfortat(Runs,Perioden,Akteur,Staerke,Nummer).
%
titfortat(R,T,AS,Staerke,Nr):-
  Vorher is T-1,
  action(R,Vorher,_,Direction,Handlungsname,Staerke,
                  Maxstaerke,Nr),
  (retract(choosed_actionart(act(_,1,_,_))); true),
  asserta(choosed_actionart(act(Handlungsname,1,
                  Maxstaerke,Nr))),
  asserta(choosed_action(act(Handlungsname,1,
                  Maxstaerke,Nr))),
  asserta(action(R,T,AS,Direction,Handlungsname,Staerke,
                  Maxstaerke,Nr)),
  feasable(action(R,T,AS,Direction,Handlungsname,Staerke,
                  Maxstaerke,Nr)),!.
```

Datei 20: tfts.

```
/*********************************************/
/***                 Regel:tfts                ***/
/*********************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: maxesk
% Diese Datei beschreibt die Regel: tfts
% tfts/5
% tfts(Runs,Perioden,Akteur,Staerke,Nummer).
%
tfts(R,T,AS,Neustaerke,Nr):-
  Vorher is T-1,
  action(R,Vorher,_,Direction,Handlungsname,Staerke,
                  Maxstaerke,Nr),
  Neustaerkeisrandom(Maxstaerke)+1,
  (retract(choosed_actionart(act(_,1,_,_))); true),
  asserta(choosed_actionart(act(Handlungsname,1,Maxstaerke,Nr))),
  asserta(action(R,T,AS,Direction,Handlungsname,
                  Neustaerke,Maxstaerke,Nr)),
  feasable(action(R,T,AS,Direction,Handlungsname,
                  Neustaerke,Maxstaerke,Nr)),!.
```

Datei 21: maxesk.

```
/*******************************************/
/***                 Regel:maxesk            ***/
/*******************************************/
%
% letzte Aenderung: 19/3/00
% Dateiname: maxesk
% Diese Datei beschreibt die Regel: maxesk
% maxesk/5
% maxesk(Runs,Perioden,Akteur,Staerke,Nummer).
%
maxesk(R,T,AS,Staerke,Nr):-
  act(grossangriff,1,1,20),
  (retract(choosed_actionart(act(_,1,_,_))); true),
  asserta(choosed_actionart(act(grossangriff,1,1,20))),
  asserta(action(R,T,AS,positiv,grossangriff,1,1,20)).
```

Datei 22: stat.prolog

```
/*********************************************/
/***                 Statistikmodul        ***/
/*********************************************/
%
%
% letzte Aenderung: 19/3/00
%
% Dateiname: stat.prolog
%
% Aufruf erfolgt mit: 'stat/0'
%
/*********************************************/
/***                    stat/0             ***/
/*********************************************/
%
%
%stat/0
%
stat:-
  get_time(X),
  statistics(cputime,CPUOLDTIME),
  statistics(inferences,Inferences),
  protocol('protokoll.txt'),
  style_check([-singleton]),
  ['prog.prolog','pred.prolog','actions.prolog',
  'feasable.prolog','rules.prolog',
  'charakters.prolog','flags.prolog'],
  del('Ergebnis'),
  consult_facts,
  asserta(runs(20)),
  asserta(periods(1000)),
  asserta(actors(2)),
  make_flags,
  asserta(actor_list([1,2])),
  asserta(use_old_data(yes)),
  use_charakter_list,
  write_flagsgesamt,
  statistik(CPUOLDTIME,Inferences),
  get_time(Y),
```

```
46   write_time(X,Y),
47   put(7),
48   protocolwrite,
49   noprotocol.
51 /*****************************************/
53 /***          use_charakter_list/0       ***/
55 /*****************************************/
57 %
58 %use_charakter_list/0
59 %
60 % Die Charaktere werden erstellt. Mit diesen Charakteren
61 % wird start/0 aus der Datei 'prog.prolog' gestartet.
62 %
63 use_charakter_list:-
64   findall(Art,charakter_arts(Art,_),Charaktere),
65   asserta(charakter_list(Charaktere)),
66   charakter_list(Charaktere),
67   permut(Charaktere,C),
68   asserta(clist(C)),
69   clist(C),
70   length(C,La),
71   (between(1,La,Cl),
72   nth1(Cl,C,Pa),
73   nth1(1,Pa,Charakter1),
74   nth1(2,Pa,Charakter2),
75   use_list(Charakter1,Charakter2),
76   fail; true,!).
77
78 use_list(Charakter1,Charakter2):-
79   Charakter1 = Charakter2,
80   asserta(staerke_list([0,10,50,75,100])),
81   staerke_list(L),
82   kart_prod(L,L,K),
83   (retract(klist(_)),fail; true),
84   asserta(klist(K)),
85   klist(K),
86   length(K,Laenge),
87   (between(1,Laenge,XX),
88   klist(K),
89   nth1(XX,K,Paar),
90   nth1(1,Paar,Staerke1),
```

```
91    nth1(2,Paar,Staerke2),
92    use_staerke_list(Charakter1,Charakter2,Staerke1,Staerke2),
93    fail; true,!).
94
95  use_list(Charakter1,Charakter2) :-
96    Charakter1\ = Charakter2,
97    asserta(staerke_list([0,10,50,75,100])),
98    staerke_list(L),
99    permut(L,K),
100   (retract(klist(_)),fail; true),
101   asserta(klist(K)),
102   klist(K),
103   length(K,Laenge),
104   (between(1,Laenge,XX),
105   klist(K),
106   nth1(XX,K,Paar),
107   nth1(1,Paar,Staerke1),
108   nth1(2,Paar,Staerke2),
109   use_staerke_list(Charakter1,Charakter2,Staerke1,Staerke2),
110   fail; true,!).
111
112 use_staerke_list(Charakter1,Charakter2,Staerke1,Staerke2) :-
113   Charakter1 = Charakter2,
114   permut([yes,no],X),
115   (retract(xlist(_)),fail; true),
116   asserta(xlist(X)),
117   length(X,Zahl),
118   (between(1,Zahl,Z),
119   xlist(X),
120   nth1(Z,X,F1),
121   nth1(1,F1,Fanatisch1),
122   nth1(2,F1,Fanatisch2),
123   use_fanatiker_list(Charakter1,Charakter2,
124                      Staerke1,Staerke2,Fanatisch1,Fanatisch2),
125   fail;
126   true,!).
127
128 use_staerke_list(Charakter1,Charakter2,Staerke1,Staerke2) :-
129   Charakter1\ = Charakter2,
130   kart_prod([yes,no],[yes,no],X),
131   (retract(xlist(_)),fail; true),
```

```
132   asserta(xlist(X)),
133   length(X,Zahl),
134   (between(1,Zahl,Z),
135   xlist(X),
136   nth1(Z,X,F1),
137   nth1(1,F1,Fanatisch1),
138   nth1(2,F1,Fanatisch2),
139   use_fanatiker_list(Charakter1,Charakter2,
140                      Staerke1,Staerke2,Fanatisch1,Fanatisch2),
141   fail;
142   true,!).
143
144 use_fanatiker_list(Charakter1,Charakter2,
145                      Staerke1,Staerke2,Fanatiker1,Fanatiker2):-
146   charakter_arts(Charakter1,Liste1),
147   charakter_arts(Charakter2,Liste2),
148   write_scharakter(Liste1,Liste2,
149                      Staerke1,Staerke2,Fanatiker1,Fanatiker2),
150   write_dcharakter(Liste1,Liste2,
151                      Staerke1,Staerke2,Fanatiker1,Fanatiker2),
152   (retract(used_rule(_)),fail; true),
153   make_runs,
154   write_flagsrun,
155   reset_flags,
156   !.
157
158 write_scharakter(Liste1,Liste2,Staerke1,Staerke2,
159                      Fanatiker1,Fanatiker2):-
160   (retract(charakter(_,_,_,_)),fail; true),
161   asserta(charakter(1,Liste1,Staerke1,Fanatiker1)),
162   asserta(charakter(2,Liste2,Staerke2,Fanatiker2)).
163
164 write_dcharakter(Liste1,Liste2,Staerke1,Staerke2,
165                      Fanatiker1,Fanatiker2):-
166   (retract(charakter(_,_,_,_)),fail; true),
167   asserta(charakter(1,Liste1,Staerke1,Fanatiker1)),
168   asserta(charakter(2,Liste2,Staerke2,Fanatiker2)),
169   calculate_sum(Liste1,Sum1),
170   calculate_sum(Liste2,Sum2),
171   write(Liste1),tab(1),write(Staerke1),
172   tab(1),write(Fanatiker1),tab(10),write(Sum1),nl,
```

```
173   tab(10),write(Sum2),tab(20),write(Liste2),tab(1),
174   write(Staerke2),tab(1),write(Fanatiker2),nl.
175
176 make_runs :-
177   del('history.prolog'),
178   del('ende.prolog'),
179   (retract(run(_)),fail; true),
180   asserta(run(1)),
181   get_old_data(1),
182   flag(lives,L,L+1),
183   begin,
184   !.
185
186 /*******************************************/
188 /***                End of file          ***/
190 /*******************************************/
192 %
193 % End of file stat.prolog
```

Datei 23: Beispiel ergebnis

```
fact([1,[0,0,0,35,0,35,0,29.99,0.01],0,no],[2,[0,0,0,35,0,35,0,29.99,0.01],0,no],[0.031941,0.029484,0.034398,
  0.039312,0.103194,0.046683,0.039312,0.00982801,0.04914,0.0810811,0.017199,0.036855,0.039312,0.022113,0.041769,
  0.00982801,0.012285,0.004914,0,0.012285,0.022113,0.0663391,0.022113,0.02457,0.034398,0.041769,0.022113,0.00982801,
  0.017199,0.041769,0,0.036855]).
fact([1,[0,0,0,35,0,35,0,29.99,0.01],0,no],[2,[0,0,0,35,0,35,0,29.99,0.01],0,yes],[0.0142119,0.00904393,
  0.00516796,0.0310078,0.152455,0.0167959,0.00904393,0.00129199,0.0232558,0.0155039,0.00516796,0.0155039,0.0180879,
  0.0142119,0.00775194,0.00258398,0,0,0,0.00516796,0.0129199,0.0116279,0.00645995,0.00904393,0.0167959,0.0142119,
  0.0116279,0.00645995,0.00904393,0.00904393,0,0.0206718]).
fact([1,[0,0,0,35,0,35,0,29.99,0.01],0,yes],[2,[0,0,0,35,0,35,0,29.99,0.01],0,yes],[0.00267278,0.00190913,
  0.00343643,0.00267278,0.602138,0.00381825,0.00649103,0.00229095,0.0045819,0.00343643,0.0030546,0.0030546,
  0.00229095,0.00229095,0.0045819,0.00343643,0.0015273,0.000381825,0.00114548,0.0076365,0.00420008,0.0045819,
  0.00267278,0.00572738,0.0015273,0.0045819,0.00229095,0.0030546,0.00687285,0.0061092,0,0]).
fact([1,[0,0,0,35,0,35,0,29.99,0.01],0,no],[2,[0,0,0,35,0,35,0,29.99,0.01],10,no],[0.00134907,0.00337268,
  0.00202361,0.00370995,0.0205734,0.00303541,0.00337268,0.000674536,0.00674536,0.00539629,0.0010118,0.00775717,
  0.00505902,0.00539629,0.00269815,0.00236088,0.000337268,0.000674536,0.000337268,0.00168634,0.00236088,0.00404722,
  0.00370995,0.0010118,0.00438449,0.00472175,0.00236088,0.00168634,0.00404722,0.00573356,0,0.00505902]).
fact([1,[0,0,0,35,0,35,0,29.99,0.01],0,no],[2,[0,0,0,35,0,35,0,29.99,0.01],10,yes],[0.00120809,0.000906071,
  0.00241619,0.00724857,0.0268801,0.00422833,0.00332226,0.000906071,0.00422833,0.00151012,0.000302024,0.00271821,
  0.00392631,0.00302024,0.00332226,0.000302024,0.000302024,0.000906071,0,0.00181214,0.0051344,0.0051344,0.00211416,
  0.00181214,0.00392631,0.00392631,0.00151012,0.000604047,0.00271821,0.00392631,0,0.00422833]).
fact([1,[0,0,0,35,0,35,0,29.99,0.01],0,yes],[2,[0,0,0,35,0,35,0,29.99,0.01],10,yes],[0.00138395,0.000395413,
  0.00059312,0.00514037,0.298142,0.0029656,0.00257019,0.000197707,0.00237248,0.00257019,0.00059312,0.00276789,
  0.00138395,0.00237248,0.00158165,0.000790826,0.000395413,0.00059312,0,0.00395413,0.00138395,
  0.00138395,0.00257019,0.000395413,0.00276789,0.00237248,0.000395413,0.00059312,0.00197707,0.000790826,0,0]).
⋮
```

Datei 24: Beispiel ende.prolog

```
% Der Run 1  ist beendet. Ende bei Periode: 17
% Akteur: 1   letzte verwendete Regel: lin2
action(1, 17, 1, positiv, grossangriff, 1, 1, 20).
% Der Run 2  ist beendet. Ende bei Periode: 31
% Akteur: 2   letzte verwendete Regel: lin2
action(2, 31, 2, positiv, grossangriff, 1, 1, 20).
% Der Run 3  ist beendet. Ende bei Periode: 8
% Akteur: 2   letzte verwendete Regel: lin2
action(3, 8, 2, positiv, grossangriff, 1, 1, 20).
% Der Run 4  ist beendet. Ende bei Periode: 1
% Akteur: 1   letzte verwendete Regel: lin2
action(4, 1, 1, positiv, grossangriff, 1, 1, 20).
% Der Run 5  ist beendet. Ende bei Periode: 40
% Akteur: 2   letzte verwendete Regel: lin2
action(5, 40, 2, positiv, grossangriff, 1, 1, 20).
% Der Run 6  ist beendet. Ende bei Periode: 44
% Akteur: 2   letzte verwendete Regel: lin2
action(6, 44, 2, positiv, grossangriff, 1, 1, 20).
% Der Run 7  ist beendet. Ende bei Periode: 2
% Akteur: 2   letzte verwendete Regel: maxesk
action(7, 2, 2, positiv, grossangriff, 1, 1, 20).
% Der Run 8  ist beendet. Ende bei Periode: 61
% Akteur: 1   letzte verwendete Regel: lin2
action(8, 61, 1, positiv, grossangriff, 1, 1, 20).
% Der Run 9  ist beendet. Ende bei Periode: 8
% Akteur: 2   letzte verwendete Regel: lin2
action(9, 8, 2, positiv, grossangriff, 1, 1, 20).
% Der Run 10  ist beendet. Ende bei Periode: 39
% Akteur: 1   letzte verwendete Regel: lin2
action(10, 39, 1, positiv, grossangriff, 1, 1, 20).
% Der Run 11  ist beendet. Ende bei Periode: 101
% Akteur: 1   letzte verwendete Regel: lin2
action(11, 101, 1, positiv, grossangriff, 1, 1, 20).
⋮
```

Datei 25: history.prolog

```
% ------------- Akteur 1 ----------
% Regeln:[linear, titfortat, tfts, desk,
                              lingr, deskgr, lin2, desk2, maxesk]
% Staerke:100  Fanatiker:yes
charakter(1,[12, 12, 12, 12, 12, 13, 13, 13.99, 0.01],100,yes).
% ------------- Akteur 2 ----------
% Regeln:[linear, titfortat, tfts, desk,
                              lingr, deskgr, lin2, desk2, maxesk]
% Staerke:100  Fanatiker:yes
charakter(2,[12, 12, 12, 12, 12, 13, 13, 13.99, 0.01],100,yes).
% action(Run, Periode, Nummerakteur, Direktion,
%              Handlungsname, Neuestaerke, Maxstaerke,
%                                          Handlungsnummer).
% Initsialisierungsperiode 0 des Runs 1 --noch keine Regel benutzt
action(1, 0, 2, positiv, manover, 2, 4, 13).
% Run: 1  Periode: 1  Akteur: 1  verwendete Regel: lingr
action(1, 1, 1, positiv, manover, 3, 4, 13).
% Run: 1  Periode: 2  Akteur: 1  verwendete Regel: lingr
action(1, 2, 1, positiv, manover, 4, 4, 13).
% Run: 1  Periode: 3  Akteur: 1  verwendete Regel: lingr
action(1, 3, 1, positiv, mobilmachung, 1, 3, 14).
% Run: 1  Periode: 4  Akteur: 2  verwendete Regel: tfts
action(1, 4, 2, positiv, mobilmachung, 3, 3, 14).
% Run: 1  Periode: 5  Akteur: 1  verwendete Regel: titfortat
action(1, 5, 1, positiv, mobilmachung, 3, 3, 14).
% Run: 1  Periode: 6  Akteur: 1  verwendete Regel: deskgr
action(1, 6, 1, positiv, mobilmachung, 2, 3, 14).
% Run: 1  Periode: 7  Akteur: 1  verwendete Regel: titfortat
action(1, 7, 1, positiv, mobilmachung, 2, 3, 14).
% Run: 1  Periode: 8  Akteur: 1  verwendete Regel: desk2
action(1, 8, 1, positiv, unterminieren, 1, 4, 26).
% Run: 1  Periode: 9  Akteur: 2  verwendete Regel: lin2
action(1, 9, 2, positiv, mobilmachung, 1, 3, 14).
% Run: 1  Periode: 10  Akteur: 2  verwendete Regel: lingr
action(1, 10, 2, positiv, mobilmachung, 2, 3, 14).
% Run: 1  Periode: 11  Akteur: 2  verwendete Regel: tfts
action(1, 11, 2, positiv, mobilmachung, 3, 3, 14).
:
```

Datei 26: tabellegesamt

```
Anzahl der ausgefuehrten Handlungen: 1158455
Anzahl der Runs:35700
 aufgeben R: 18904
 grossangriff R: 16796
 Max. Anz. Perioden: 0
angriffsdrohung_0positiv, 41351, 23.1658, 1.15829
beschuldigung_0positiv, 32426, 18.1658, 0.908291
vergeltung_0positiv, 33668, 18.8616, 0.943081
sanktionen_0positiv, 29846, 16.7204, 0.836022
miskredit_0positiv, 268478, 150.408, 7.52039
erschweren_0positiv, 34638, 19.405, 0.970252
handel_0positiv, 56264, 31.5204, 1.57602
unterdrueckung_0positiv, 26011, 14.572, 0.728599
verunglimpfung_0positiv, 28081, 15.7317, 0.786583
embargo_0positiv, 41442, 23.2168, 1.16084
blockade_0positiv, 22392, 12.5445, 0.627227
zurueckhalten_0positiv, 31708, 17.7636, 0.888179
manover_0positiv, 38573, 21.6095, 1.08048
mobilmachung_0positiv, 30551, 17.1154, 0.85577
truppenverlegung_0positiv, 28554, 15.9966, 0.799832
grenzueberschreitung_0positiv, 23126, 12.9557, 0.647787
luftraum_0positiv, 18066, 10.121, 0.50605
schiffeversenken_0positiv, 13089, 7.33277, 0.366639
bombardieren_0positiv, 11392, 6.38207, 0.319104
grossangriff_0positiv, 16796, 9.40952, 0.470476
waffenproduktion_0positiv, 37499, 21.0078, 1.05039
konstruktion_0positiv, 36095, 20.2213, 1.01106
anlagenbau_0positiv, 29112, 16.3092, 0.815462
legitimation_0positiv, 29272, 16.3989, 0.819944
aufruf_0positiv, 34579, 19.372, 0.968599
unterminieren_0positiv, 28477, 15.9535, 0.797675
sabotage_0positiv, 37008, 20.7328, 1.03664
terror_0positiv, 27937, 15.651, 0.782549
demonstration_0positiv, 27155, 15.2129, 0.760644
streik_0positiv, 25965, 14.5462, 0.727311
skip_0positiv, 0, 0, 0
aufgeben_0positiv, 18904, 10.5905, 0.529524
```

Datei 27: Beispiel ergebnis_neu

```
num([1,[92,1,1,1,1,1,1,1,1.99,0.01],0,no],[2,[92,1,1,1,1,1,1,1,1.99,0.01],0,no],0.082902,0.041451,0.067357,0.056995,
  0,0.025907,0.062176,0,0.031088,0.036269,0.025907,0.005181,0.067357,0.025907,0.031088,0.031088,0.041451,0.020725,
  0.010363,0.010363,0.005181,0.036269,0.025907,0.015544,0.046632,0.041451,0.005181,0,0.036269,0.020725,0,0.093264).
num([1,[92,1,1,1,1,1,1,1,1.99,0.01],0,no],[2,[92,1,1,1,1,1,1,1,1.99,0.01],0,yes],0.057143,0.038095,0.028571,
  0.016667,0,0.007143,0.011905,0.038095,0.014286,0.011905,0.021429,0.016667,0.016667,0.009524,0.014286,0.007143,
  0.007143,0,0.004762,0.007143,0.004762,0.016667,0.040476,0.016667,0.009524,0,0.038095,0.038095,0.007143,
  0,0,0.040476).
num([1,[92,1,1,1,1,1,1,1,1.99,0.01],0,yes],[2,[92,1,1,1,1,1,1,1,1.99,0.01],0,yes],0.036350,0.021513,0.037834,
  0.011128,0.002967,0.002967,0.007418,0.017804,0.018546,0.033383,0.031157,0.012611,0.025223,0.029674,0.027448,
  0.025964,0.031157,0.020030,0.044510,0.014837,0.017804,0.025964,0.022255,0.012611,0.023739,0.031157,0.022997,
  0.027448,0.020030,0.031899,0,0).
num([1,[92,1,1,1,1,1,1,1,1.99,0.01],0,no],[2,[92,1,1,1,1,1,1,1,1.99,0.01],10,no],0.001329,0.002658,0.003322,0.003987,
  0.003987,0.004651,0.005980,0.000664,0.003322,0.001329,0,0.009967,0.007973,0,0,0.002658,0.006645,0.002658,
  0,0.001329,0.001329,0.005980,0.004651,0.002658,0.003322,0.003322,0.001993,0,0.005980,0.000664,0,0.011960).
num([1,[92,1,1,1,1,1,1,1,1.99,0.01],0,no],[2,[92,1,1,1,1,1,1,1,1.99,0.01],10,yes],0,0,0.001153,0.006916,0.002882,
  0.001729,0.005764,0.008069,0.004035,0.008069,0.008646,0.004611,0.001153,0.005764,0.005764,0.003458,0.001153,
  0.001729,0.003458,0.002305,0.001729,0,0,0.002305,0.005764,0.007493,0.008069,0.011527,0.005187,0.004611,
  0,0.009222).
num([1,[92,1,1,1,1,1,1,1,1.99,0.01],0,yes],[2,[92,1,1,1,1,1,1,1,1.99,0.01],10,yes],0.020882,0.011601,0.019722,
  0.004640,0,0.000773,0.002707,0.015081,0.004640,0.014695,0.024749,0.005414,0.005027,0.010054,0.011601,0.017401,
  0.020108,0.015855,0.024362,0.007734,0.008121,0.010441,0.008507,0.007347,0.005800,0.006961,0.013534,0.020495,
  0.005027,0.005800,0,0).
⋮
```

Datei 28: maxende.prolog

```
% Der Run 1  wurde abgebrochen.
% Ende bei Periode: 1000 ----
                        Maximale Anzahl der Perioden erreicht.
% Akteur: 2   letzte verwendete Regel: desk2
% letzte Handlung:
action(1, 1000, 2, positiv, miskredit, 1, 3, 5).
Charakter der Akteure:
% ------------  Akteur 1  ------------
% Regeln:[linear, titfortat, tfts, desk,
                        lingr, deskgr, lin2, desk2, maxesk]
% Staerke:0  Fanatiker:yes
charakter(1,[17, 85, 1, 1, 1, 1, 1, 1.99, 0.01],0,yes).
% ------------  Akteur 2  ------------
% Regeln:[linear, titfortat, tfts, desk,
                        lingr, deskgr, lin2, desk2, maxesk]
% Staerke:75  Fanatiker:yes
charakter(2,[10, 20, 10, 9, 4, 1, 1, 44.99, 0.01],75,yes).

% Der Run 13  wurde abgebrochen.
% Ende bei Periode: 1000 ----
                        Maximale Anzahl der Perioden erreicht.
% Akteur: 2   letzte verwendete Regel: desk2
% letzte Handlung:
action(13, 1000, 2, positiv, miskredit, 1, 3, 5).
Charakter der Akteure:
% ------------  Akteur 2  ------------
% Regeln:[linear, titfortat, tfts, desk,
                        lingr, deskgr, lin2, desk2, maxesk]
% Staerke:10  Fanatiker:yes
charakter(2,[10, 20, 10, 9, 4, 1, 1, 44.99, 0.01],10,yes).
% ------------  Akteur 1  ------------
% Regeln:[linear, titfortat, tfts, desk,
                        lingr, deskgr, lin2, desk2, maxesk]
% Staerke:0  Fanatiker:yes
charakter(1,[17, 85, 1, 1, 1, 1, 1, 1.99, 0.01],0,yes).
⋮
```

Datei 29: protokoll.sve

```
1  /*******************************************/
3  /***                Akteure              ***/
5  /*******************************************/
7  %
9  [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 no
10                [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 no
11 [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 no
12                [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 yes
13 [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 yes
14                [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 yes
15 [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 no
16                [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 10 no
17 [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 no
18                [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 10 yes
19 [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 yes
20                [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 10 yes
21 [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 0 no
22                [0, 0, 0, 35, 0, 35, 0, 29.99, 0.01] 50 no
23 ⋮
24 [30, 2, 2, 2, 30, 2, 30, 1.95, 0.05] 0 no
25                [12, 13, 13, 13, 12, 12, 12, 12.99, 0.01] 50 yes
26 [30, 2, 2, 2, 30, 2, 30, 1.95, 0.05] 0 no
27                [12, 13, 13, 13, 12, 12, 12, 12.99, 0.01] 50 no
28 [30, 2, 2, 2, 30, 2, 30, 1.95, 0.05] 0 yes
29                [12, 13, 13, 13, 12, 12, 12, 12.99, 0.01] 75 yes
30 [30, 2, 2, 2, 30, 2, 30, 1.95, 0.05] 0 yes
31                [12, 13, 13, 13, 12, 12, 12, 12.99, 0.01] 75 no
32 [30, 2, 2, 2, 30, 2, 30, 1.95, 0.05] 0 no
33                [12, 13, 13, 13, 12, 12, 12, 12.99, 0.01] 75 yes
34 [30, 2, 2, 2, 30, 2, 30, 1.95, 0.05] 0 no
35                [12, 13, 13, 13, 12, 12, 12, 12.99, 0.01] 75 no
36 [30, 2, 2, 2, 30, 2, 30, 1.95, 0.05] 0 yes
37                [12, 13, 13, 13, 12, 12, 12, 12.99, 0.01] 100 yes
38 [30, 2, 2, 2, 30, 2, 30, 1.95, 0.05] 0 yes
39                [12, 13, 13, 13, 12, 12, 12, 12.99, 0.01] 100 no
40 ⋮
```

Abbildungen

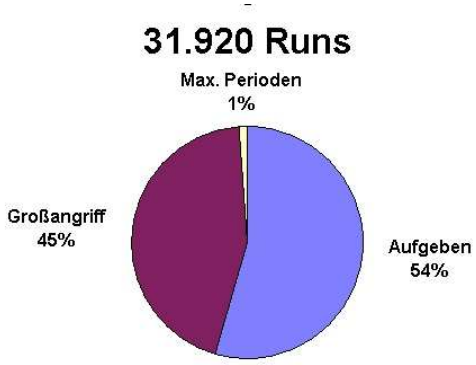

Abbildung 18: Verteilung des Simulationsendes bei 31.920 Runs und 1.000 Perioden

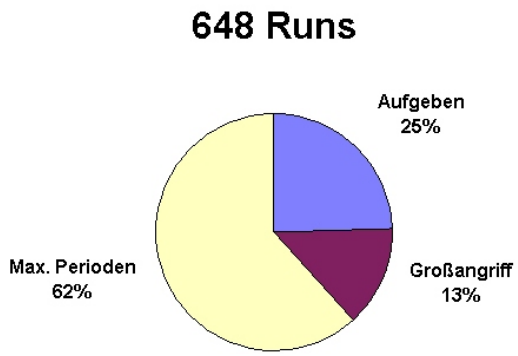

Abbildung 19: Verteilung des Simulationsendes bei einem Run und 200 Perioden

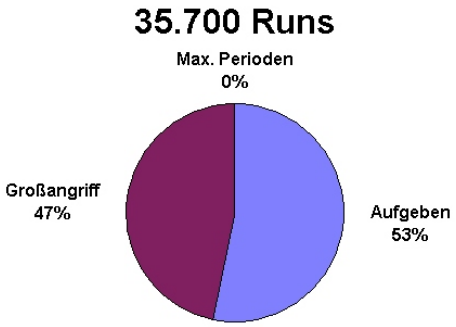

Abbildung 20: Verteilung des Simulationsendes bei 35.700 Runs und 1.000 Perioden

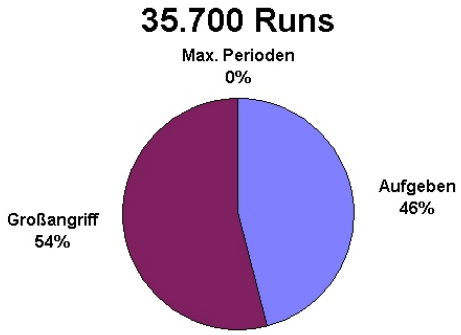

Abbildung 21: Verteilung des Simulationsendes bei 35.700 Runs und 1.000 Perioden

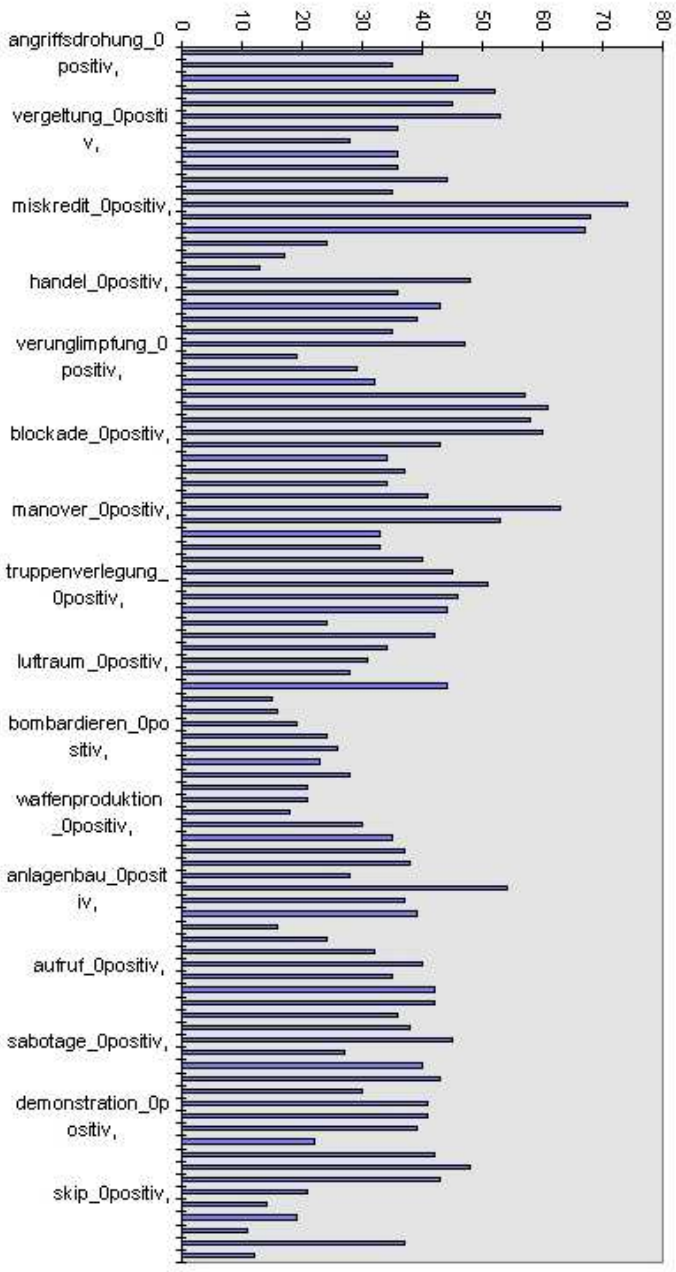

Abbildung 22: Verteilung der Handlungen I
bei 100 Perioden und zwei Handlungsgruppen.

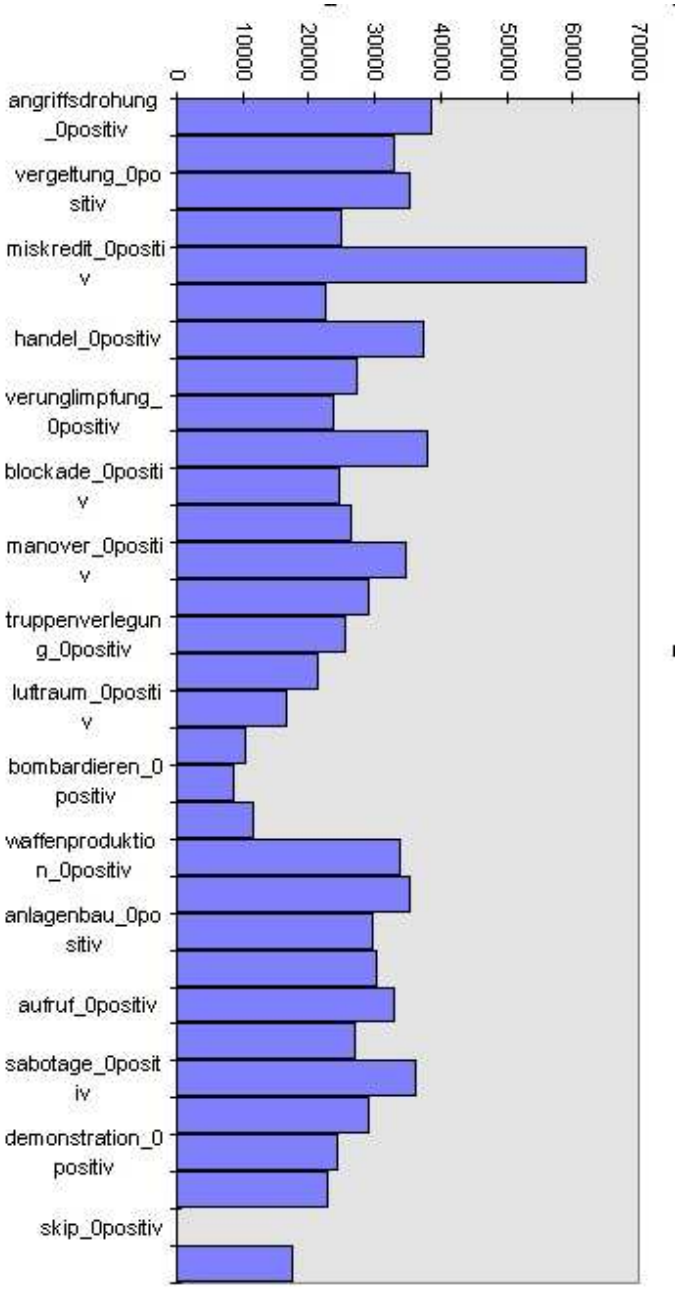

Abbildung 23: Verteilung der Handlungen II

bei 100 Perioden und zehn Handlungsgruppen.

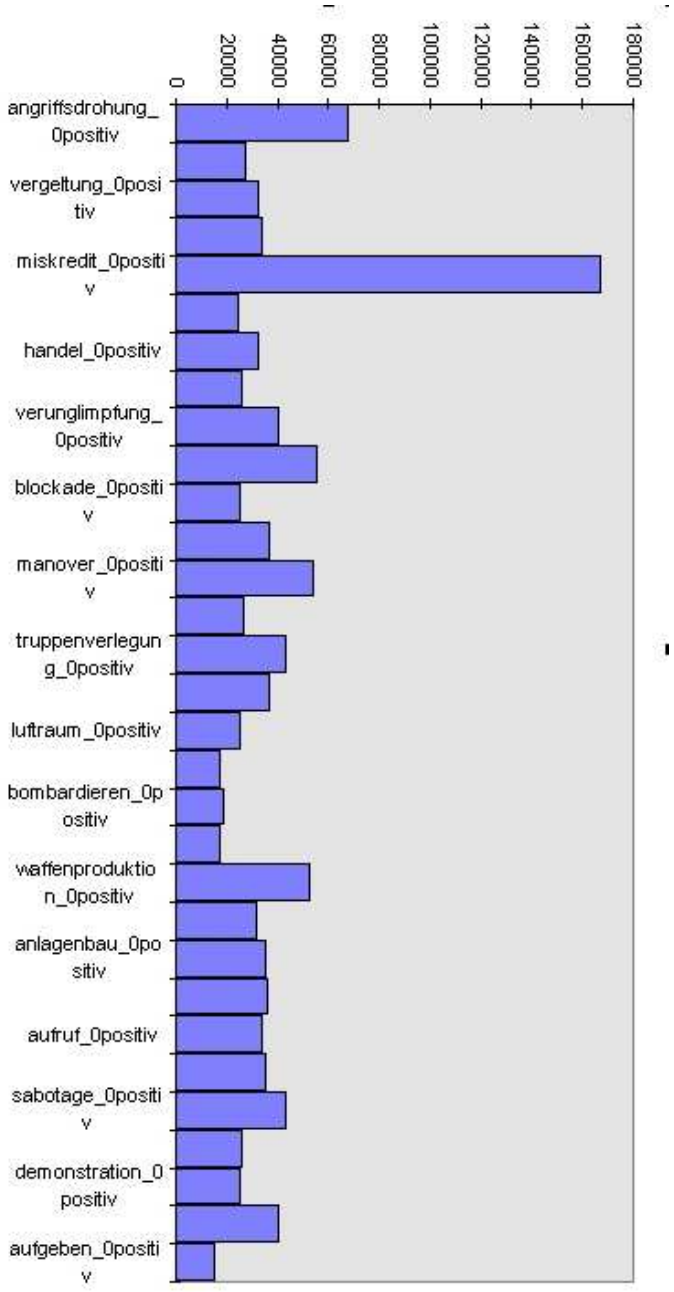

Abbildung 24: Verteilung der Handlungen III bei 1.000 Perioden und zehn Handlungsgruppen.

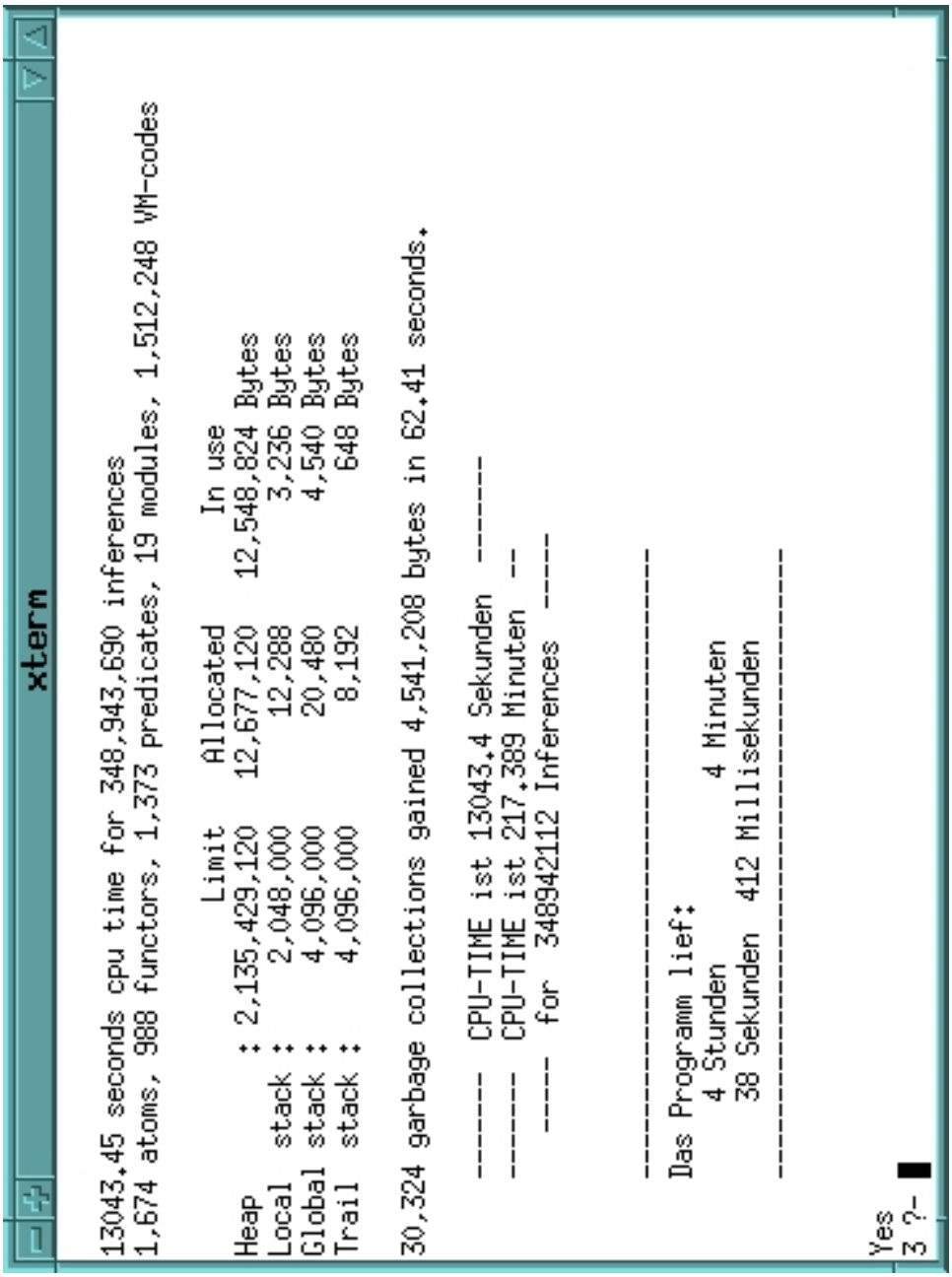

Abbildung 25: Statistik des Prologlaufs.

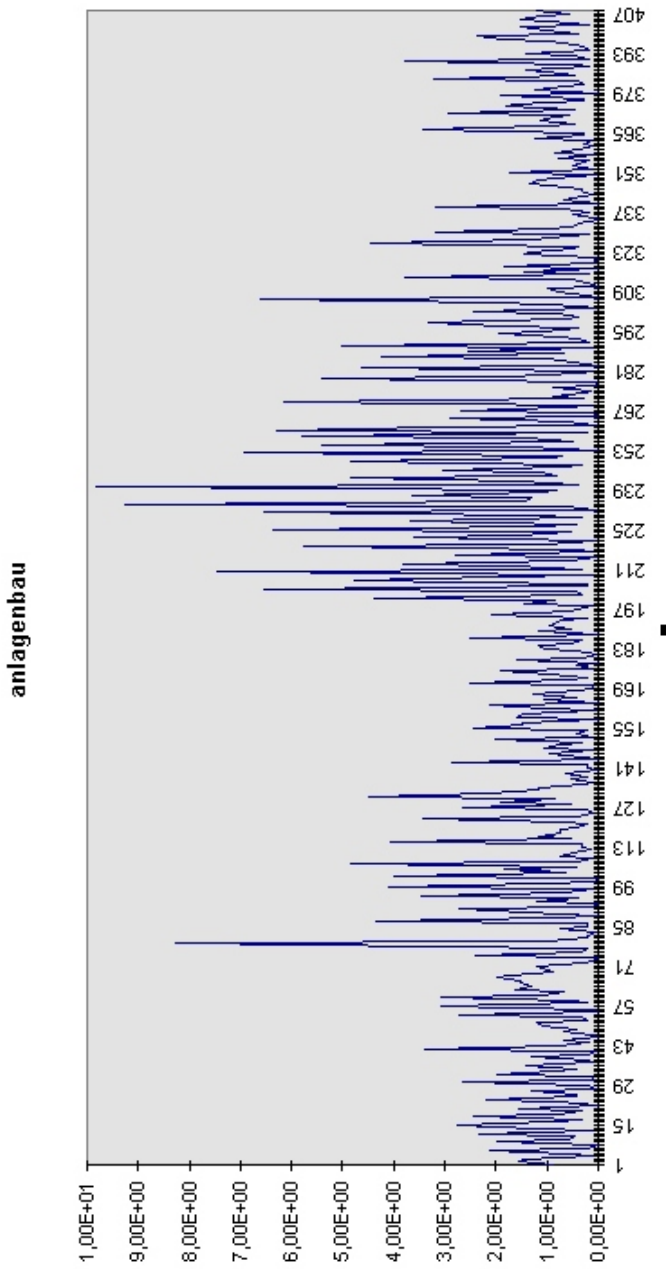

Abbildung 26: Verteilung der Handlung `anlagenbau`

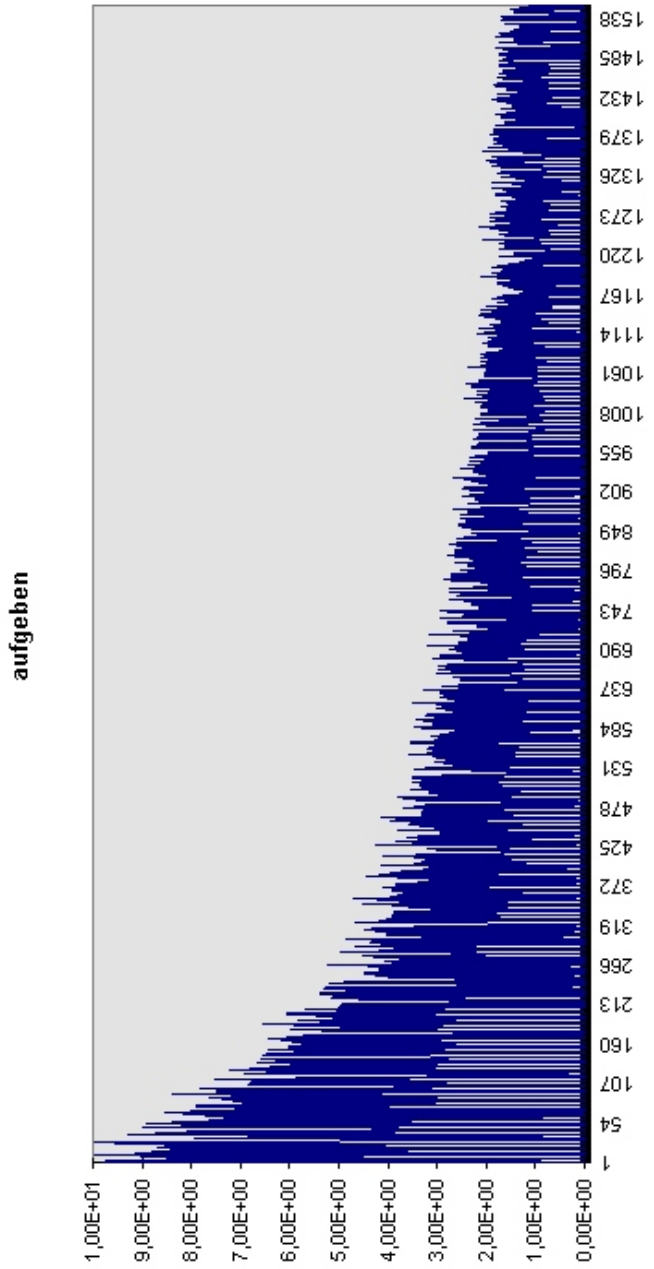

Abbildung 27: Verteilung der Handlung `aufgeben`

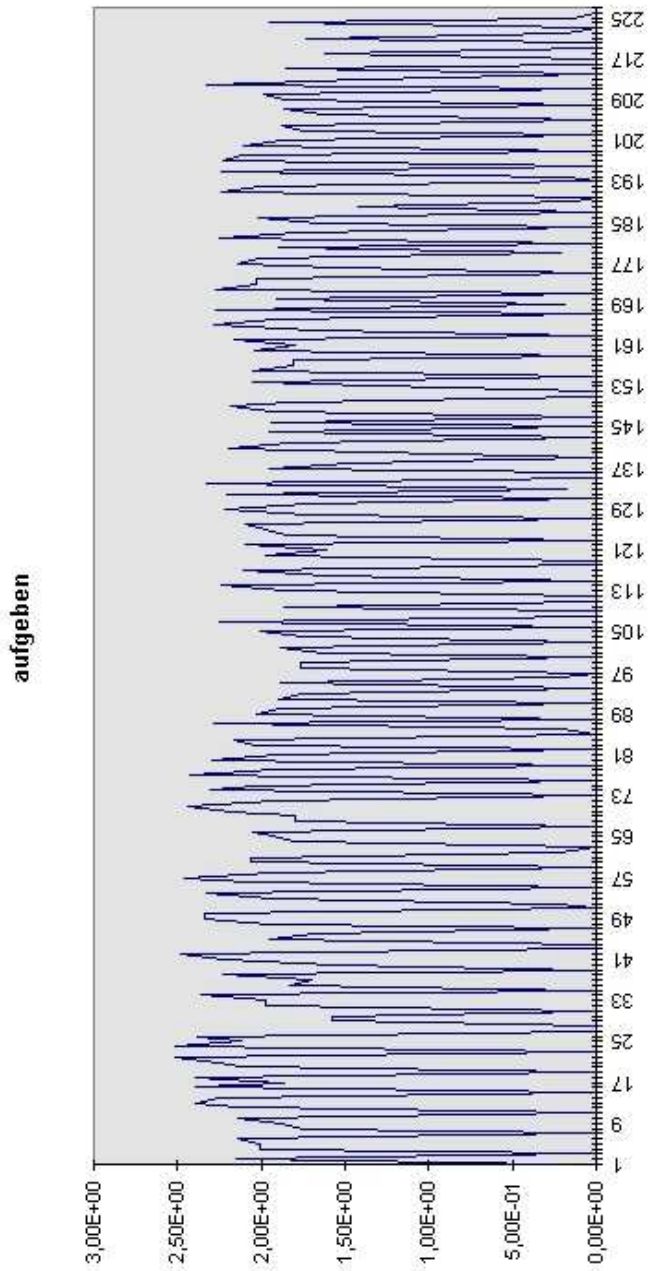

Abbildung 28: Verteilung der Handlung `aufgeben` bei gleichen Charakteren

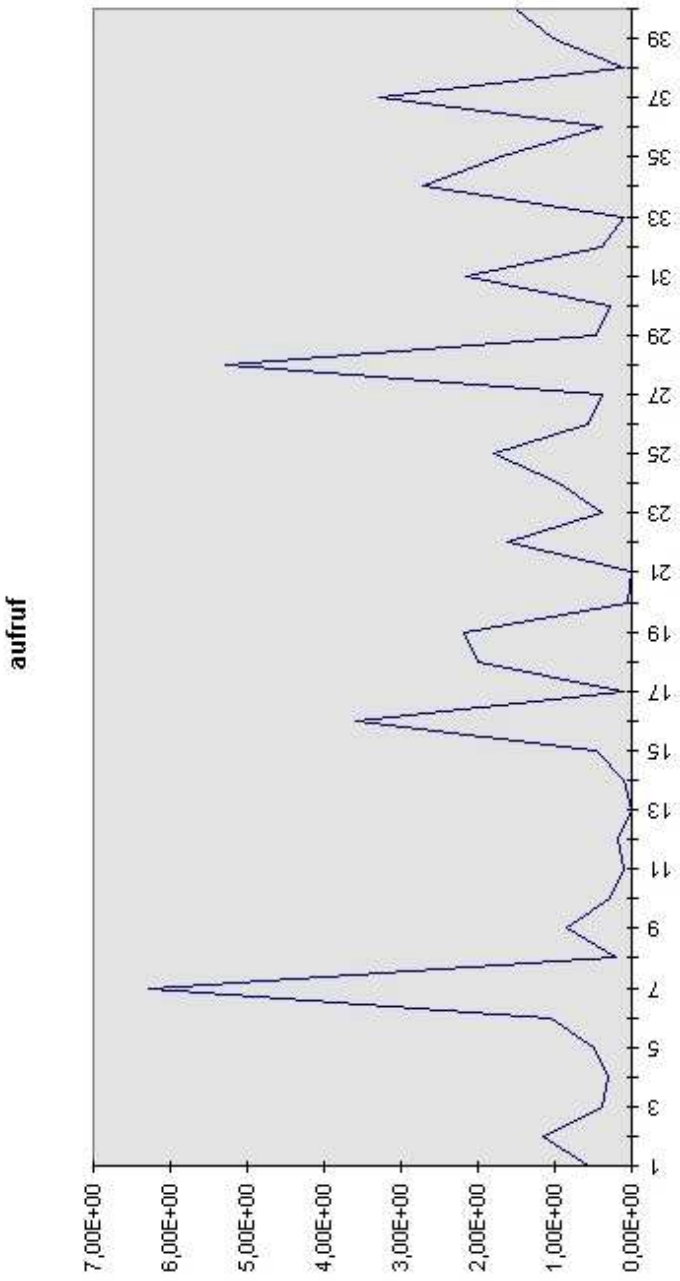

Abbildung 29: Verteilung der Handlung `aufruf`

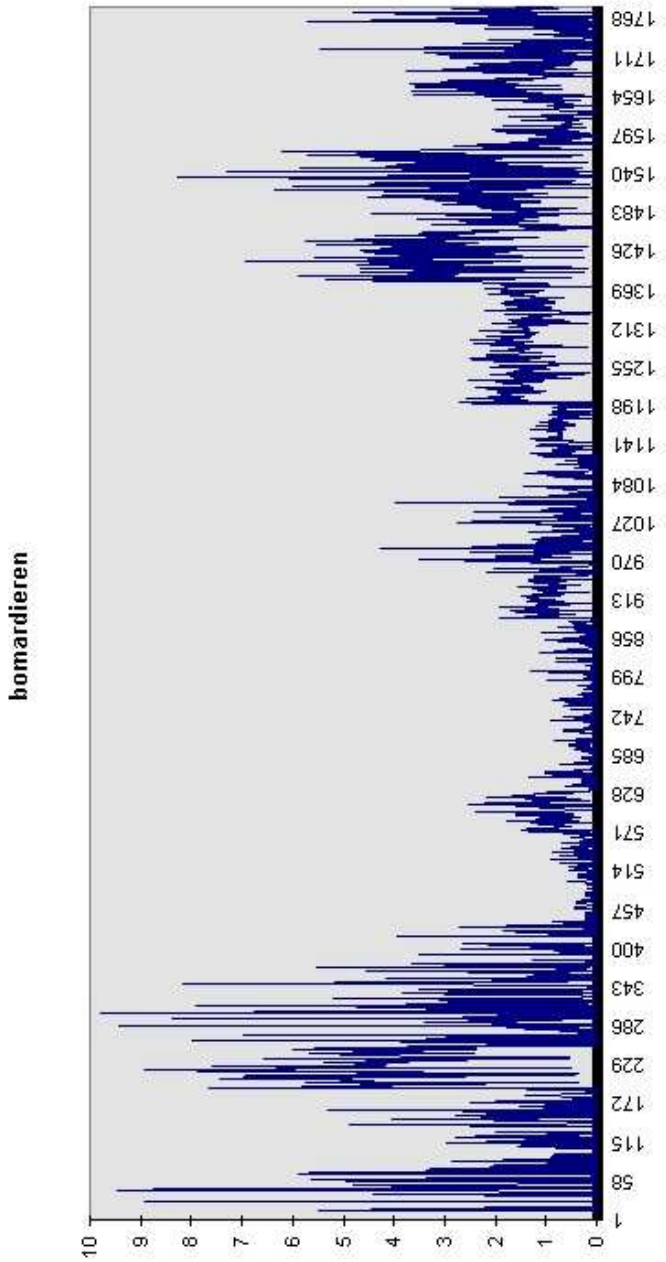

Abbildung 30: Verteilung der Handlung `bombardieren` bei Charakterverteilung nach Datei 92cha.prolog.

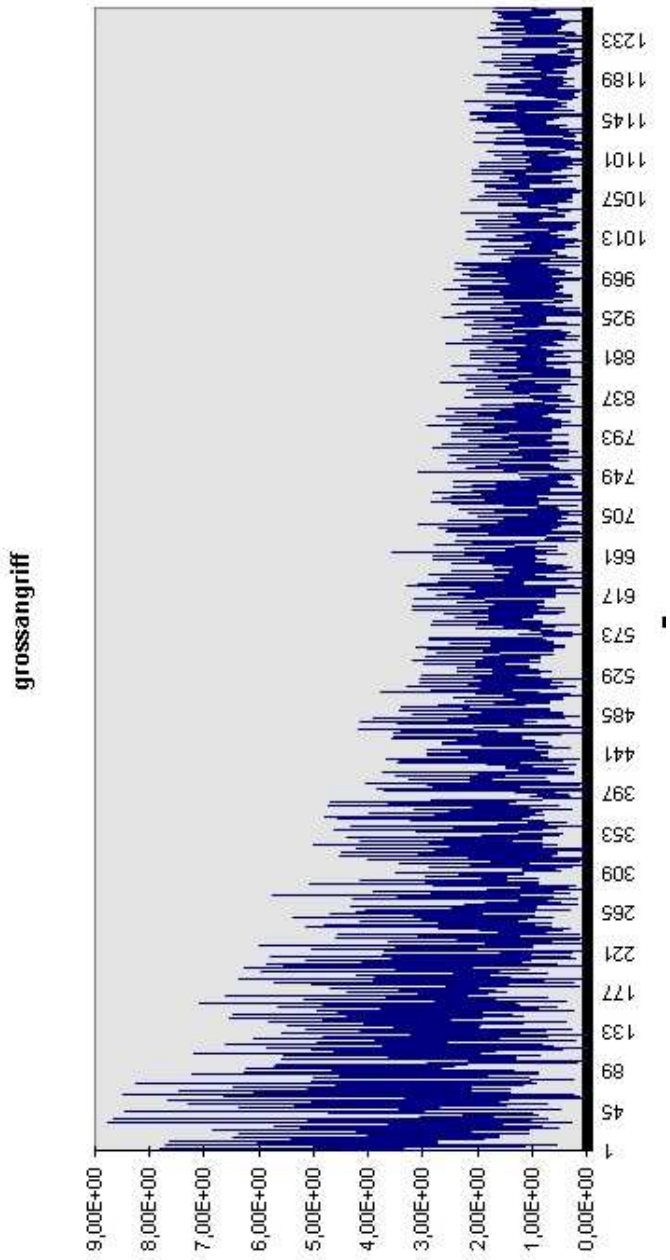

Abbildung 31: Verteilung der Handlung `grossangriff`

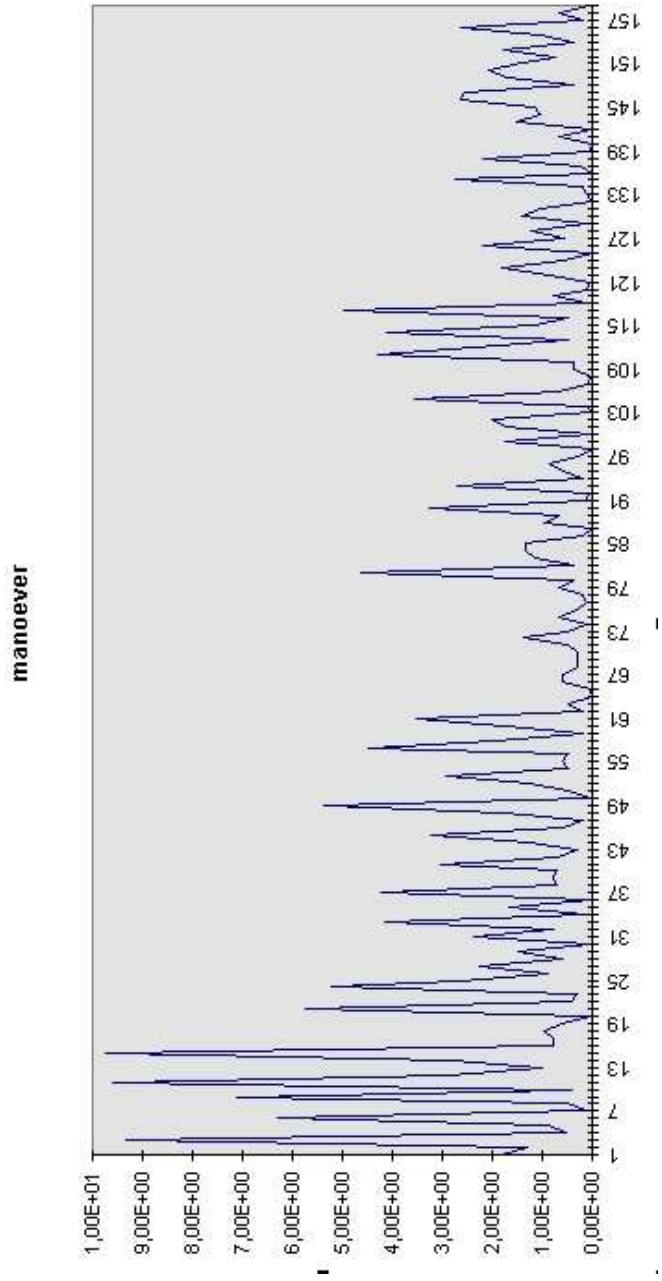

Abbildung 32: Verteilung der Handlung `manoever`

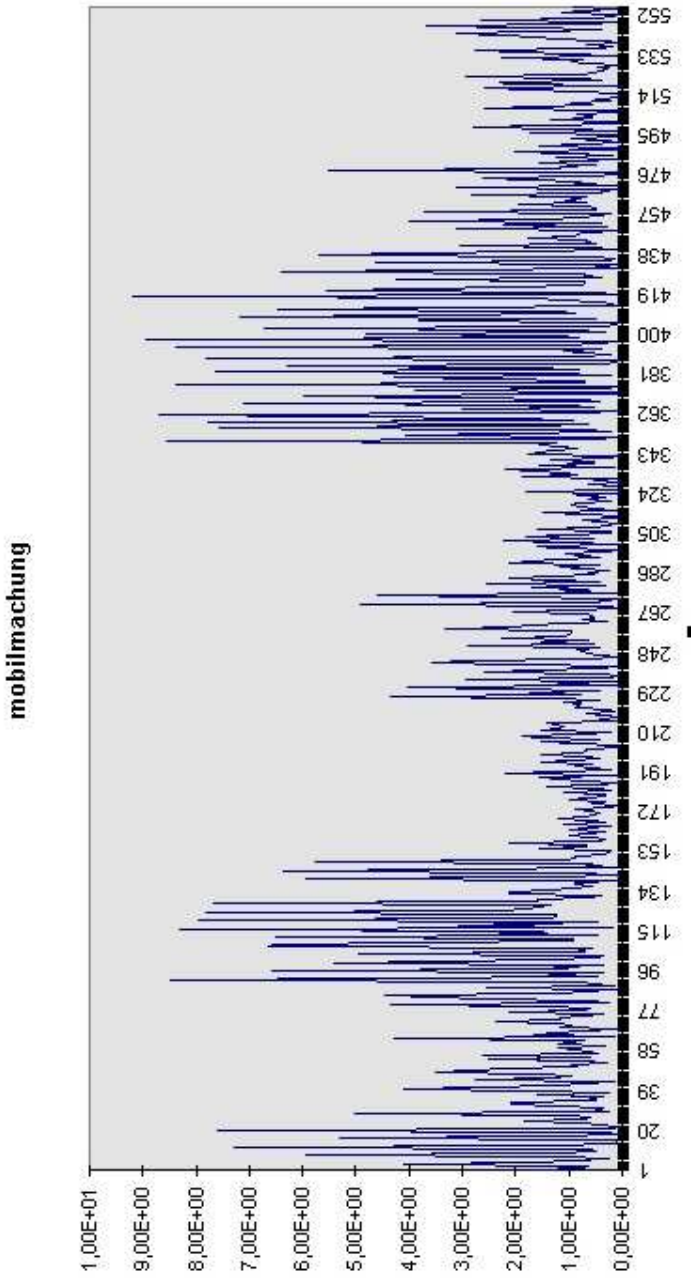

Abbildung 33: Verteilung der Handlung `mobilmachung`

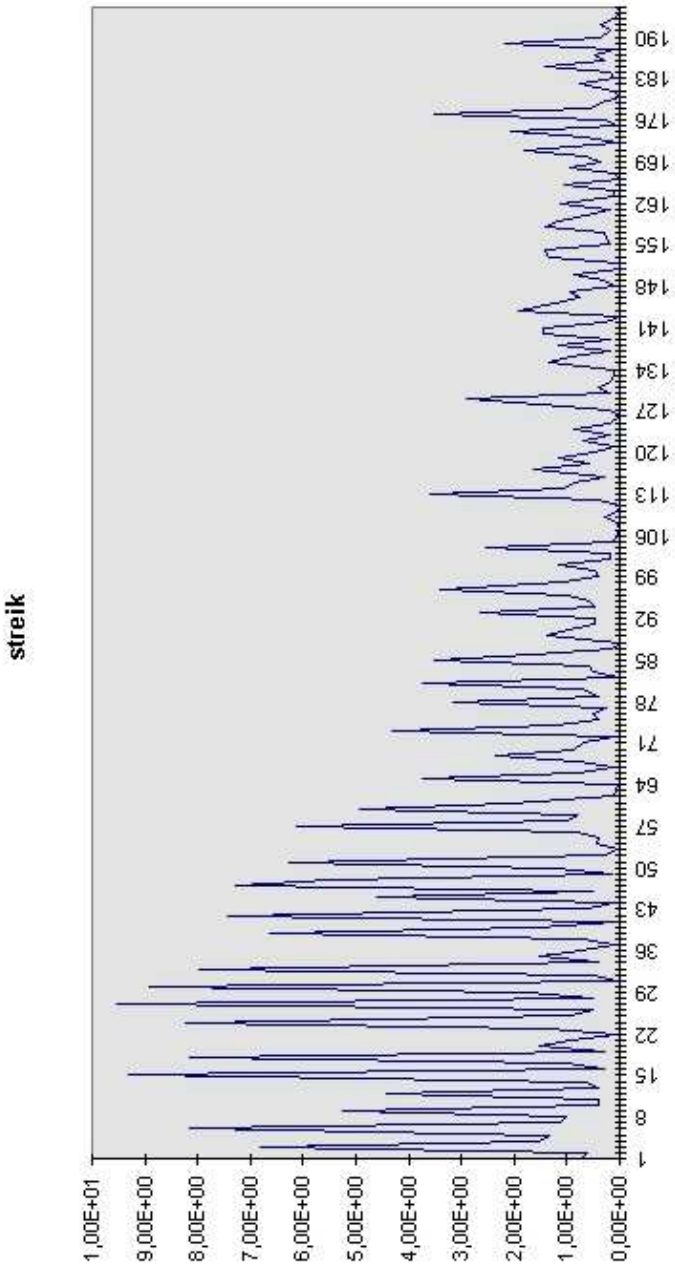

Abbildung 34: Verteilung der Handlung `streik`

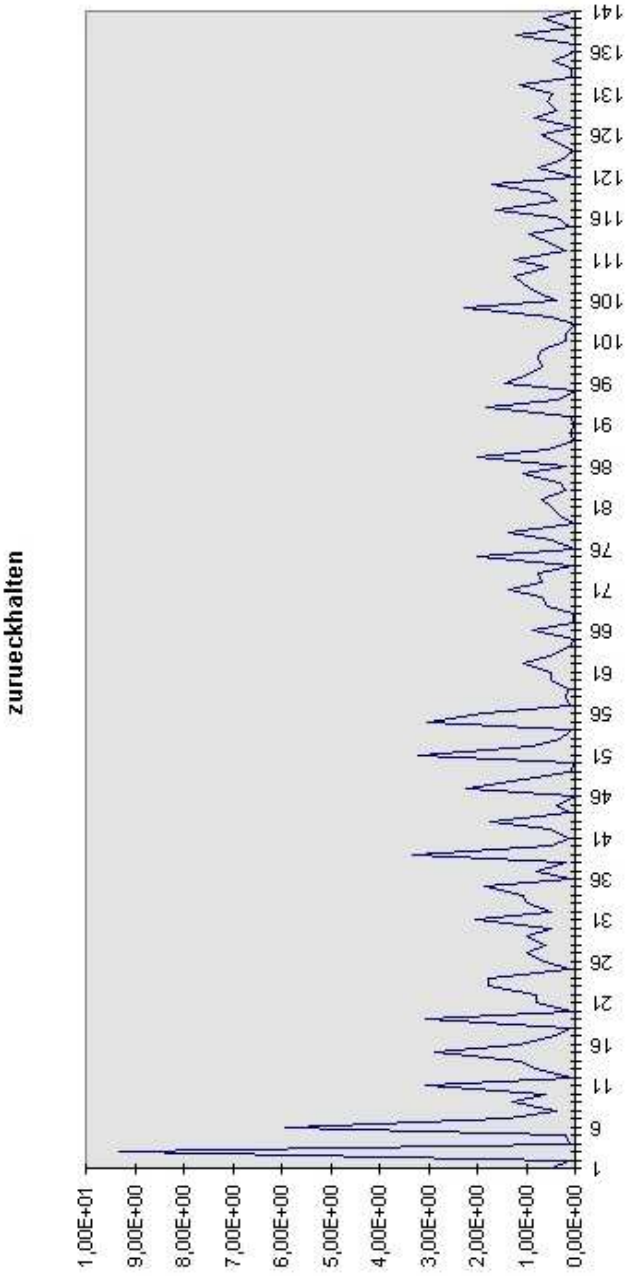

Abbildung 35: Verteilung der Handlung `zurueckhalten`

Verzeichnis der Personen

Verzeichnis der Symbole

Verzeichnis der Prologprädikate

Stichwörter

Anlagen

Der Arbeit ist eine CD-Rom beigefügt. Auf der CD befinden sich drei Verzeichnisse:

- Text,
- Programm und
- Daten.

Im Verzeichnis Text befinden sich zwei Versionen der Arbeit im Postscriptformat:

- twoside.ps und
- oneside.ps.

Wie die Namen schon ausdrücken ist eine Version einseitig, die andere doppelseitig. Im Verzeichnis Programm befinden sich alle Programme. Die Programme sind im Anhang auf Seite XXX bis Seite CVI auch ausgedruckt. Unter Daten sind einige Programmläufe abgespeichert. Da ein Programmlauf zwischen zwei und sechs Stunden auf einem 200 MHZ Rechner benötigt, ist für eine Kontrolle der Ergebnisse ein fertiger Lauf sicher sinnvoll.